UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Estudio del ransomware en dispositivos móviles Android: análisis y contramedidas

Albert Soler Colomé
Tutor: David Arroyo Guardeño

Mayo 2017

Estudio del ransomware en dispositivos móviles Android: análisis y contramedidas

AUTOR: Albert Soler Colomé
TUTOR: David Arroyo Guardeño

Grupo de Neurocomputación Biológica
Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Mayo de 2017

Resumen (castellano)

Este Trabajo Fin de Grado consiste en el desarrollo de un sistema para gestionar y realizar copias de seguridad (*backups*) de dispositivos móviles Android en el seno de una organización. Está formado por un servidor programado en Python con Flask, una aplicación web y una aplicación cliente para Android desarrollada en Java.

La motivación del proyecto viene dada por el auge del *ransomware* destinado a los dispositivos móviles Android. *Ransomware* es un término inglés que denota un tipo particular de virus informático. Se caracteriza por secuestrar un recurso y pedir un pago como rescate para liberarlo. Existen dos tipos principales, los primeros te encriptan los archivos personales con una clave aleatoria desconocida para el usuario. De esta manera los datos quedan inaccesibles hasta que se paga el rescate. La segunda vertiente bloquea el acceso al dispositivo. En el caso particular de los móviles, cambiando el patrón de desbloqueo y superponiendo una pantalla de bloqueo para que no se pueda realizar ninguna acción con el dispositivo. Para poder retomar el control del terminal se debe pagar el rescate que piden los cibercriminales. Para evitar ser descubiertos, los malhechores piden el pago a través de una criptodivisa pseudo-anónima como el Bitcoin o mediante tarjetas de prepago.

Existen diversos mecanismos para evitar o mermar los daños ocasionados por el *ransomware*. La primera opción sería no instalar o acceder a fuentes de información no debidamente verificadas. Al igual que detectarlo antes de que penetre en el sistema, desgraciadamente esto no es siempre factible. Dado que no podemos excluir fallas en las recomendaciones anteriores, conviene contar con un buen sistema de copias de seguridad que contengan todos los archivos previos al ataque. Para de este modo, poder recuperar el estado del sistema. Es por ello que nosotros nos centraremos en ese aspecto, en facilitar la creación y subida de copias de seguridad de los móviles a nuestro servidor. Se hará uso de nuestro servidor para habilitar el análisis posterior de los datos y la realización de auditorías o análisis forenses de ellos en caso de que se produzca alguna incidencia.

El proyecto ha sido realizado con la seguridad en mente desde el análisis y diseño, es por ello que todas las comunicaciones van cifradas gracias a SSL. Además, la autenticación e ingreso en las aplicaciones es a través de verificación en dos pasos, ya que se necesita la contraseña del usuario y un código generado aleatoriamente a través de una aplicación de generación de contraseñas de un solo uso.

La solución propuesta en este proyecto proporciona un servidor, una aplicación web y un cliente móvil para la creación de políticas de *backups* de dispositivos Android. Los usuarios podrán registrarse y dar de alta sus dispositivos móviles, de forma que recibirán notificaciones cada vez que sea preciso realizar el *backup* de cada uno de dichos dispositivos. Esta solución por tanto se podría englobar en un sistema MDM (*Mobile Device Management*) para el control de dispositivos móviles.

Palabras clave (castellano)

Secuestro de información, Android, Virus, Software maligno, Software malicioso, Flask, MDM, Aplicación web, Firebase, Software malicioso, Heldroid, API REST.

Abstract (English)

This Bachelor Thesis consists in the development of a backup system for Android mobile phones. It is intended to use within an organization, where the users can upload their phone's data. It is formed by a Server programmed in Python with the Flask framework, a web application and a client android application programmed in Java.

The main motivation for developing this project comes from the boom of Android ransomware. Ransomware is a type of malicious software that blocks access to the victim's data or device until a ransom is paid. In particular, the Android ransomware blocks the access to the device by changing its pin number for unblocking the screen. This type is called lock-screen ransomware, and the ones that encrypt your personal files are called "crypto-ransomware". The perpetrators want to remain anonymous so they demand the payment of the ransom to be made through pseudo-anonymous cryptocurrency like Bitcoin. Although there have been reports of Android ransomware requesting the payment with prepaid cards.

There are different systems and measures to avoid or reduce the ransomware attack damages. The first option would be not installing or accessing information sources which are not duly verified. Just like being able to detect the malware before it infects the phone; unfortunately this is not always feasible. Since we cannot rule out flaws in previous recommendations it is advisable to have a good backup system up to date which contains all the files previous to the attack. Having this backup would allow a complete restore on the system. That is why we will focus on that: facilitate the creation and upload off backups from the Android phones to our server. We will use our server to enable the subsequent analysis of the data and the performance of audits or forensic analysis of them in the event of any occurrence.

This project has been developed with security in mind; security-by-design policy. From the early stages of development: analysis and design. That is why all the communications are encrypted using SSL. In addition, authentication and entry into applications is through two-step verification, as the user's password and a randomly generated code are required. The code is generated through a one-time passwords generation application.

The solution proposed in this project provides a server, a web application and a mobile client for the creation of backup policies for Android devices. Users will be able to register and register their mobile devices, so that they will receive notifications every time it is necessary to back up each of these devices. This solution could therefore be encompassed in an MDM (Mobile Device Management) system for the control of mobile devices.

Keywords (inglés)

Ransomware, Android, Virus, Malware, Flask, MDM, Web application, Firebase, Heldroid, Malicious software, API REST.

Agradecimientos

Me gustaría expresar mi agradecimiento a todas las personas que me han ayudado y apoyado en la realización de este TFG.

En particular, me gustaría destacar la labor de David Arroyo, mi tutor, por brindarme la oportunidad de realizar un trabajo sobre un tema actual y de gran importancia, que satisface mi interés personal y profesional. En segundo lugar, agradecerle su implicación, su disponibilidad total y sus constantes consejos y sugerencias para mejorar el presente trabajo. Por último, me gustaría destacar su labor explicándome conceptos y la importancia de la seguridad informática y la privacidad.

A todos mis compañeros y amigos que me han apoyado y animado. Es especial, a mi familia y pareja, sin ellos esto no hubiera sido posible.

A todos los demás, que de alguna manera habéis estado implicados en el desarrollo de este proyecto, gracias de corazón.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

El auge en los ataques de *ransomware* en las empresas es evidente, como pudimos comprobar el pasado viernes 12 de mayo de 2017 con el ataque masivo a escala global protagonizado por el ransomware llamado WannaCry [1]. Afectó a multitud de países como podemos ver en la Figura 1, atacando a empresas tan diversas como compañías de telecomunicaciones españolas, hospitales en Reino Unido y compañías de reparto, como FedEx en Estados Unidos [2].

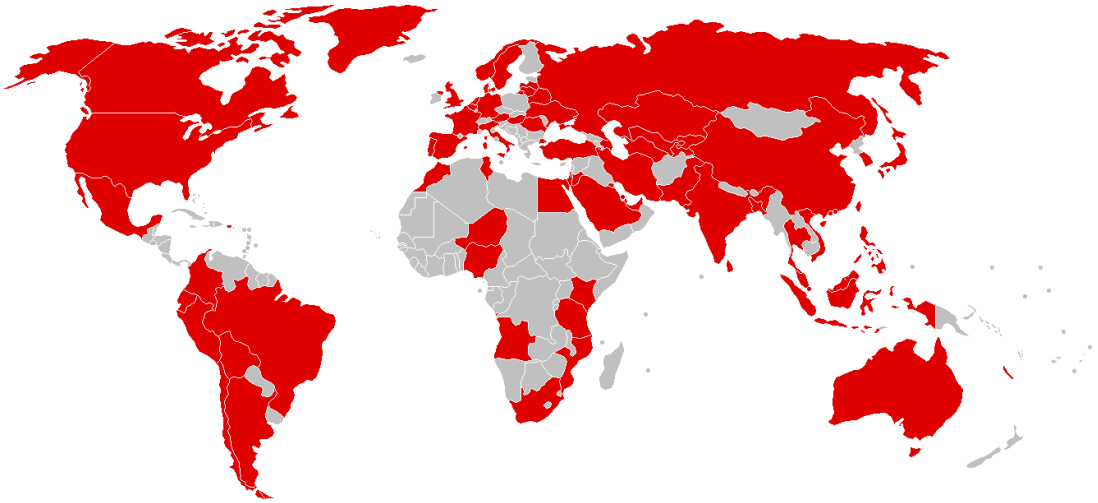

Figura 1: Países afectados en las primeras horas del ciber-ataque de Wannacry (en rojo), según fuentes del laboratorio de investigación de Kaspersky. Extraído de [2]**.**

En este caso se aprovechaba de una vulnerabilidad del sistema operativo Windows [3], aunque el *ransomware* no afecta exclusivamente a este sistema operativo. En específico, es de especial relevancia el despliegue de *ransomware* sobre dispositivos Android, ya que éste es el sistema operativo más usado en la actualidad [4]. Es por ello que es necesario entender el funcionamiento del *ransomware* en estos dispositivos para poder prevenirlo y combatirlo. Así como diseñar estrategias de control y recuperación de daños una vez los dispositivos han sido afectados. Por lo tanto, crear una solución que nos permita recuperar los datos después del ataque es la primera motivación del presente trabajo. Además de tener nosotros el control de los datos de la copia de respaldo.

La segunda motivación tiene carácter personal. En efecto, este trabajo de fin grado ha tenido también como objetivo profundizar en el conocimiento del campo de la ciberseguridad, y ha permitido poner en práctica los conocimientos adquiridos en las diferentes asignaturas de la carrera. También ha hecho factible estudiar y aplicar diferentes tecnologías que no habían sido utilizadas previamente, como es el caso de Flask, además de propiciar una maduración en el grado de conocimiento de Python y de programación de aplicaciones móviles en Android. Por último, pero no menos importante, este trabajo fin de grado ha exigido desarrollar un proyecto completo, diseñándolo desde el principio con la seguridad como un requisito fundamental. Es decir, siguiendo el criterio de *security-by-design* [5].

1.2 Objetivos

Los objetivos del trabajo se pueden dividir en dos grupos, primero los de investigación y posteriormente de desarrollo.

Para la parte de investigación se ha realizado un estudio acerca del *ransomware* destinado a Android, estudiando sus vectores de infección, sus características principales, las diferentes familias y ramas de este tipo de *malware*, el impacto que tiene y su funcionamiento.

En el caso de los dispositivos móviles hay dos grandes familias de *ransomware*, las de bloqueo del móvil y las que cifran los datos. Frente a este ataque existen diferentes contramedidas, principalmente:

- Control orientado a la detección, es decir, anticipar el ataque mediante el análisis del comportamiento estándar del mismo antes de su despliegue total. Un ejemplo sería la herramienta HelDroid [6], [7]. Gracias a la cual se ha realizado un análisis de diferentes muestras de *ransomware* que se detallarán en el apartado 2.3.

- Control recuperativo, permite restablecer el estado del sistema al estado previo del ataque. En la línea temporal del ataque se sitúa al final, después de que este haya sucedido. Un ejemplo de este control son las copias de respaldo o de seguridad. Que nos permite una vez el ataque ha tenido éxito, recuperar todos los archivos a través de una copia de seguridad o *backup*[1]. Esta solución es válida para cualquier tipo de *ransomware.*

Posteriormente, para la parte de desarrollo se ha construido una herramienta de apoyo a un administrador de sistemas para realizar *backups* de los móviles Android de la empresa. En particular, se ha **desarrollado** un **servidor** y **aplicación web** desde la cual los usuarios se pueden registrar y realizar diversas opciones de configuración. Y los administradores también para poder gestionar a los usuarios. Después se ha creado una **aplicación cliente en Android** que permite a un usuario crear una política de *backups* con frecuencias de subida al servidor, por ejemplo: cada 8 días debe subir el *backup* que contiene los documentos de las reuniones semanales. Y cada día debe subir el *backup* con datos más críticos. Aparte de crearlos, desde la aplicación podrá listarlos, modificarlos, eliminarlos, y como es lógico, subirlos al servidor. Es importante destacar que los *backups* no son gestionados por ninguna tercera parte como Android Backup Service, lo que nos otorga un control total sobre ellos.

Por lo tanto, se ha desarrollado un conjunto de herramientas que no existían en el mercado. Los beneficios de utilizar nuestro propio servidor son diversos. En primer lugar tenemos un control de privacidad de los datos respaldados que nos permite asegurar la confidencialidad e integridad de los mismos. Además al poseer los datos podemos implementar estrategias de *data loss prevention* (DLP) [8] y realizar análisis y estadísticas de los mismos. Así como realizar auditorías o análisis forenses de los datos en caso de que se produzca alguna incidencia.

[1] Aunque en castellano existen los términos copia de respaldo o copia de seguridad, para facilitar la lectura y escritura, y dado que *backup* es un término altamente aceptado, en el resto del documento se utilizará el término *backup* en lugar de hablar de copia de respaldo.

Al haber creado nosotros las herramientas, podemos adaptarlas a los requisitos particulares de la organización y realizar las modificaciones pertinentes cuando sea necesario.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Estado del Arte:** En el capítulo 2 se explica que es el Ransomware y la importancia que tiene en nuestro día a día. Posteriormente nos centraremos en el ransomware en Android que es el foco del presente documento. Explicando su funcionamiento y características principales así como dar algún ejemplo.

- **Análisis:** El capítulo 3 se realizará el análisis de requisitos, definidos según los objetivos deseados en las aplicaciones finales y delimitados por el alcance del proyecto. Tanto del Servidor, como de la aplicación Web y aplicación Android. Se describe la funcionalidad a partir del catálogo de requisitos y casos de uso.

- **Diseño:** En el capítulo 4 se detallará la fase de diseño, considerando las estructuras de datos empleadas, el flujo de navegación de la aplicación, el mapa web, y las maquetas de las aplicaciones web y Android, además de comentar detalles estructurales de la arquitectura.

- **Desarrollo:** En el capítulo 5 se explica el método de desarrollo llevado a cabo, junto con las características Software y Hardware del equipo de desarrollo, y las herramientas y lenguajes utilizados. Explicando el porqué de su elección.

- **Integración, pruebas y resultados:** En el capítulo 6 se detallan las pruebas unitarias de integración realizadas así como los resultados obtenidos. Mostrando capturas de pantalla de las aplicaciones y explicando el proceso de integración.

- **Conclusiones y trabajo futuro:** Finalmente, en el capítulo 7 se enumeran las conclusiones obtenidas tras la realización del presente proyecto y se menciona el trabajo futuro a realizar para mejorarlo y aumentar su funcionalidad.

2 Estado del arte

El *ransomware* es un tipo de software malicioso (*malware*[2]) que tiene como objetivo forzar a los usuarios a pagar un rescate para recuperar el acceso de su sistema [9]. *Ransom* significa rescate en inglés. Hay dos tipos principales: los primeros te bloquean el acceso al sistema o dispositivo y la segunda variante te encripta los ficheros personales para que no puedas leerlos o acceder a ellos. Es decir, te "secuestran" un recurso, el acceso al dispositivo y/o tus datos.

El pago se realiza normalmente a través de Bitcoin o tarjetas prepago como MoneXy, MoneyPak o QIWI [10] para que sea más difícil rastrear a los atacantes. Ya que no están asociados a su nombre, se podrían considerar pseudo-anónimos porque están relacionados con un pseudónimo [11].

2.1 Importancia del ransomware

El *ransomware* está cobrando mucha importancia al tratarse de un negocio muy lucrativo, del orden de mil millones de dólares al año según fuentes del FBI [12]. Este dinero es obtenido a través de los pagos del rescate, y podemos observar un rápido crecimiento de los 24 millones de dólares pagados en el 2015. Además, se sospecha que la nube puede ser el próximo objetivo de estos ataques dado que es un nuevo foco por explotar [13].

Un debate muy interesante es si se debe pagar o no el rescate. Pagar implica financiar directamente a los cibercriminales con las consecuencias que ello conlleva: que se realices más ataques por que les son rentables y puedan mejorar los ataques y comprobar vulnerabilidades y *exploits* para abarcar un mayor número de víctimas. Además hay cibercriminales que exigen diferentes cantidades de dinero por el rescate en función del país, 700$ en Estados Unidos de media frente a los $500 en México, Israel o Rusia [14].

Para evitar ser infectados por el *ransomware* lo mejor es evitar acceder a fuentes de información que no sean debidamente verificadas. Como esto no siempre es posible, es necesario tener un mecanismo de salvaguarda. En nuestro caso, el principal mecanismo de salvaguarda será el *backup* y una correcta política de gestión de los respaldos. En particular, para el *ransomware* el *backup* debe estar fuera del contexto de la operación del *malware*. Si no es así, también se cifraría el *backup*.
Es por ello que en el presente documento nos centraremos en crear una solución que permita tener varios *backups* actualizados de sus teléfonos móviles en un servidor y fácilmente configurables por el usuario.

En lo referente al ransomware orientado a dispositivos móviles y en particular en España, el 1,25% de los usuarios se han visto afectados por este virus según un informe de Vodafone [15]. No ha sido hasta 2015 que se ha empezado a extender a los móviles coincidiendo con la generalización de su uso como medio principal para el acceso a internet y consumo de contenidos multimedia.

[2] *Malware* es un anglicismo cuyo término en castellano sería programa maligno. Dado que es un término altamente aceptado, para facilitar la lectura y escritura del presente documento será el término que se utilizará.

2.2 Ransomware en Android

Los programadores de *malware* están adaptando al *ransomware* móvil las técnicas más efectivas del *ransomware* en ordenadores. Para dificultar su detección y maximizar sus ingresos.

Las pantallas de bloqueo falsas, que evitan que puedas usar el dispositivo hasta que pagues el rescate, suelen ser variantes del "*ransomware* de la policía" [16]. Variante que intenta infundir miedo en las víctimas al acusarlas (falsamente) de diferentes delitos como el de posesión de material ilegal.

También podemos ver como al igual que la familia de *ransomware* de Windows CryptoLocker se mejoró con primitivas criptográficas más resistentes y mejor implementadas, el *crypto-ransomware* (i.e. *ransomware* que cifra los archivos) está utilizando criptografía "fuerte" lo que impide (salvo fallo en la implementación) la recuperación de los documentos y archivos sin pagar. Y como se ha comentado anteriormente, los móviles están creciendo como fuente consumo de contenidos multimedia, así como de producción, donde mucha información del día a día, como fotografías, documentos, están almacenadas en ellos y no en nuestros ordenadores. Este factor es conocido por los cibercriminales, de forma que el *ransomware* móvil es una amenaza para la integridad y disponibilidad de la información sensible almacenada en dispositivos móviles.
Una observación que puede sorprender es el cambio demográfico de las víctimas, ya que el centro de atención ya no es Europa del Este. Muchas familias recientes de *ransomware*, como Android/Simplocker y Android/Lockerpin están atacando principalmente a Estados Unidos [10].

2.2.1 Vectores comunes de infección

Como la mayoría del *malware* de Android, el *ransomware* suele estar presente en un caballo de Troya (enmascarado como una aplicación legítima). Normalmente suelen disfrazarse de aplicaciones relacionadas con la pornografía, con los juegos o las apuestas. Este tipo de aplicaciones no están disponibles en la tienda oficial de Android, Google Play para poder eludir los controles de seguridad. Son descargadas desde otras tiendas no oficiales, *torrents* o directamente desde una página web.

En algunas ocasiones, la aplicación solo contiene el logo (icono) y nombre y descripción de la aplicación legítima, mientras que otras veces está añadida en el código de la aplicación legítima, de esta manera la aplicación tiene la funcionalidad esperada por el usuario y no levanta sospechas.

Los programadores de *malware* están empezando a utilizar otras técnicas para infectar los dispositivos como utilizar otra aplicación que lo descargue. También encriptar el *payload*[3] malicioso. Así como moverlo a las carpetas de recursos de Android, destinadas a imágenes y contenidos [17].

[3] *Payload* es un término inglés que se utiliza para denominar la parte del *malware* que ejecuta la acción maligna. Como encriptar los datos, bloquear el dispositivo o enviar spam.

2.2.2 Command and control communication

La mayoría de *ransomwares* después de instalarse de manera satisfactoria se conectan con un servidor *Command and control*[4] (en adelante C&C). En ocasiones simplemente para poder llevar un registro de la infección (enviando información del dispositivo Android como el modelo, el número IMEI, el idioma, versión de Android, y otra información que les pueda resultar útil a los cibercriminales).

Por otro lado, hay veces que establecen un canal de comunicaciones permanente entre el servidor C&C y el dispositivo para que este pueda recibir órdenes (escuchar y ejecutar comandos) creando una *botnet* de dispositivos infectados bajo el control del cibercriminal.
Aparte de los comandos típicos del *ransomware* como cifrar, bloquear el dispositivo y mostrar el mensaje con la información del rescate, estos soportan varios más como:

- Borrar el contenido del dispositivo (formatearlo).
- Reinicializar el PIN de bloqueo.
- Abrir URL arbitrarias en el navegador del móvil.
- Enviar un mensaje SMS a cualquier contacto o a todos ellos.
- Bloquear o desbloquear el dispositivo.
- Robar los mensajes SMS recibidos.
- Robar los contactos del móvil.
- Mostrar otro mensaje con información del secuestro.
- Actualizar el *ransomware* a una nueva versión.
- Habilitar o deshabilitar los datos móviles.
- Habilitar o deshabilitar la conexión wifi.
- Obtener la localización GPS.

Normalmente el protocolo de comunicación utilizado es el HTTP, pero también se han identificado casos de *malware* que se comunican con el servidor de C&C vía Google Cloud Messaging[5] (GCM). Un protocolo similar también utilizado es el Baidu Cloud Push.
Algunas versiones más sofisticadas han utilizado protocolos como XMPP (Jabber) o algunos más seguros a través de dominios de Tor.onion.
Alternativamente algunos *ransomwares* pueden recibir comandos y enviar datos a través de los SMS.

2.2.3 Auto-protección en el Ransomware

Infectar un dispositivo con *malware* no es una tarea sencilla debido a las medidas de seguridad de Android y que el usuario puede tener instaladas soluciones *anti-malware* o antivirus. Es por ello que una vez consigue instalarse el *ransomware* intenta permanecer el máximo de tiempo en el móvil.

Para lograrlo pueden utilizar diversas técnicas, por ejemplo Android Lockerpin intenta eliminar los procesos que pertenezcan a aplicaciones antivirus. Aunque una de las técnicas más utilizadas es la de obtener permisos de Administrador de dispositivos (no debemos confundir con permisos de *root*, que son más peligrosos).

[4] *Command and control* se refiere a una infraestructura que entre otros elementos contiene servidores y tiene como fin controlar el *malware*, enviándole órdenes.

[5] Google Cloud Messagin es un servicio que a través de servidores de Google permite enviar y recibir datos desde la aplicación a tu servidor.

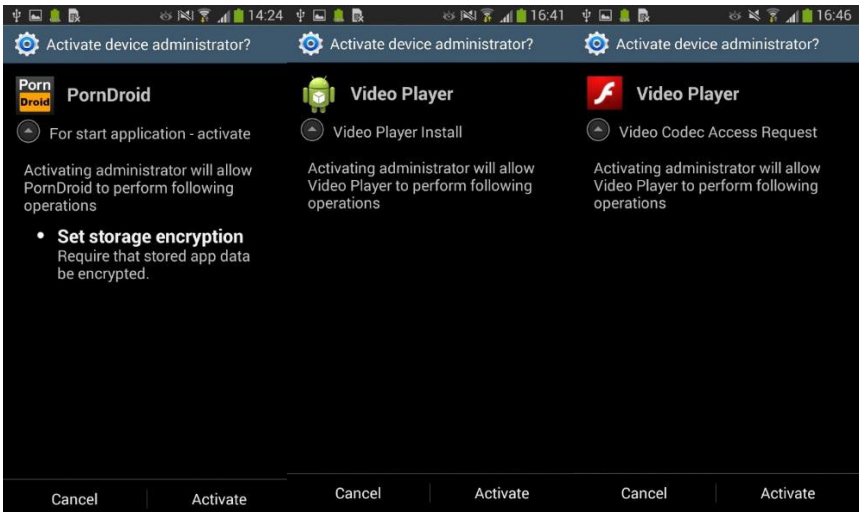

Figura 2: Ejemplos de *Malware* de Android solicitando permisos de Administrador de Dispositivos (Extraídas de [5]).

Para obtenerlos utilizan una técnica conocida como *click jacking* o *tap jacking*. Técnica que consiste en crear dos capas superpuestas. Una falsa que se muestra al usuario y una por debajo que habilita dar a la aplicación los permisos de Administrador de dispositivos.
Al pulsar en la visible, el usuario también pulsa (sin su consentimiento o conocimiento) en la invisible situada debajo de la visible aumentando los permisos y posibilidades del *ransomware*.

A continuación veremos un ejemplo del *ransomware* Android.Lockdroid.E [18], [19].
Cuando la aplicación maliciosa (una aplicación falsa para ver pornografía) es instalada y ejecutada por el usuario, se llama al diálogo de sistema para aceptar darle los permisos de Administrador de dispositivos que es ocultada por una ventana falsa de "Instalación de Paquetes" (ver Figura 3).

Figura 3: Falso diálogo de instalación (Extraída de [7]).

Mientras se muestra el diálogo de la Figura 3, en segundo plano la aplicación está encriptando todos los ficheros localizados en la memoria externa y obteniendo la información sensible del usuario almacenada en el teléfono.

Cuando el botón de continuar (Continue) se pulsa, la aplicación solicita la API del administrador de dispositivos. Normalmente el diálogo de sistema de confirmación es la capa superior de la interfaz de usuario. Pero en este caso el *ransomware* utiliza una ventana de tipo TYPE_SYSTEM_ERROR para que se muestre por encima que la del sistema (se puede ver en la Figura 4). A pesar de ser una ventana de TYPE_SYSTEM_ERROR está diseñada para que parezca un mensaje de espera ya que los componentes se están desempaquetando. Cuando en realidad nada está sucediendo y es el *malware* que está esperando unos segundos sin hacer nada.

Figura 4: Falso diálogo de desempaquetar componentes.

Finalmente después de la falsa espera, se muestra un diálogo de instalación completada. Es en este punto dónde el *malware* engaña a los usuarios para que le den los permisos. El diálogo de instalación completada es en realidad una ventana de tipo TYPE_SYSTEM_OVERLAY, utilizan esta ventana ya que tiene como característica que no puede tomar ningún tipo de foco de entrada [20]. Esto implica que la ventana no puede responder a las pulsaciones de los botones, pero la ventana debajo de esta -en este caso el diálogo de administración de dispositivos- si puede. Como se observa en la Figura 5 donde el botón continuar (Continue) está situado perfectamente encima del botón activar (Activate) lo que significa que al pulsar continuar, en realidad están pulsando activar.

Figura 5: Ejemplo de superposición, *ClickJacking*.

La buena noticia es que esto solo afecta a versiones de Android inferiores a Android 5.0 (Lollipop) ya que a partir de esa versión evitan que este tipo de ventanas puedan sobreponerse a los diálogos de sistema. El problema es que esto afecta a casi el 32 por ciento de dispositivos Android [21].

El *ransomware* está camuflado como una aplicación de pornografía llamada Porn'O'Mania. Aplicación que no se encuentra en la tienda oficial Google Play, pero puede ser descargada desde otras tiendas de aplicaciones, foros y *torrents*. Los usuarios que tienen la tienda oficial instalada Google Play están a salvo gracias a Verify Apps a pesar de que la descarguen desde otro sitio.

2.3 Estudio del ransomware Android con HelDroid

Para analizar APKs y comprobar si contienen *ransomware* antes de instalarlas en el dispositivo se ha utilizado la herramienta HelDroid [7].
Una vez instalada la herramienta se ha utilizado el siguiente comando:

```
java -jar build/libs/heldroid-all.jar detector scan
test2\b103f3897b1619dee157e62a1737e864452a85bab613ad971ac6193b3f6a4834.
apk test2\output3.csv test2\
```

La APK analizada corresponde con el malware DSEncrypt [21]. Al ejecutar el análisis obtenemos el resultado en formato (.json) y en formato (.csv). El contenido es idéntico en ambos ficheros con la salvedad que en el (.csv) un campo es la ruta de la APK. Y en el (.json) su propio nombre es el nombre de la APK.
A continuación se copia el contenido del fichero: b103f3897b1619dee157e62a1737e864452a85bab613ad971ac6193b3f6a4834.json


```
{
  "lockDetected": true,
  "lockStrategy": "DrawOver",
  "textDetected": false,
  "textScore": 0,
  "languages": [],
  "hasRWPermission": true,
  "encryptionDetected": false,
  "photoCaptureDetected": false,
  "deviceAdminUsed": true,
  "deviceAdminPolicies": "[USES_POLICY_FORCE_LOCK]",
  "fromReflection": false,
  "textComment": "null",
  "suspiciousFiles": "null"
}
```


Como se puede observar, gracias al campo: "lockDetected" se trata de un *malware* que bloquea la pantalla.

Si utilizamos el siguiente comando:

```
java -jar build/libs/heldroid-all.jar detector scan test2\
2fcd8c40e3b59786a2661054bcc2ee4124a80aee737035f59995a943b29302fd.apk.ap
k test2\output2.csv test2\
```

Obtendremos el siguiente fichero 2fcd8c40e3b59786a2661054bcc2ee4124a80aee737035f59995a943b29302fd.json:


```
{
  "lockDetected": true,
  "lockStrategy": "DrawOver",
  "textDetected": true,
  "textScore": 1,
  "languages": [
    "english"
  ],
  "hasRWPermission": true,
  "encryptionDetected": false,
  "photoCaptureDetected": false,
  "deviceAdminUsed": true,
  "deviceAdminPolicies": "[USES_ENCRYPTED_STORAGE]",
  "fromReflection": false,
  "textComment": "Threat: 1.000000, Porn: 1.000000, Law: 0.732520, Copyright: 0.348155, Moneypak:
0.894427",
  "suspiciousFiles": "{
                        copyright = [res/values/strings.xml, assets/home.html],
                        law = [res/values/strings.xml, assets/home.html],
                        moneypak=[res/values/strings.xml, assets/home.html],
                        threat=[res/values/strings.xml, assets/home.html],
                        porn=[res/values/strings.xml, assets/home.html]}"
}
```


En este caso obtenemos más información acerca del *ransomware* analizado. Por ejemplo el campo "textDetected" nos indica que ha detectado un texto que se muestra. Información que conjuntamente al campo de "lockDetected" nos indica la presencia de un *ransomware* que nos bloquea la pantalla y nos muestra una pantalla para pagar el rescate.
"textComment" nos indica el tipo de texto que muestra, en este caso se trata de una amenaza principalmente relacionada con la pornografía y la ley. Dónde el método de pago parece ser MoneyPak. Es muy probable que estemos por lo tanto ante alguna versión del "virus de la policía".

3 Análisis

A lo largo de esta sección se procederá a realizar un análisis del sistema propuesto, detallando sus objetivos principales y su alcance. Además, se incluirá la lista completa de requisitos funcionales y no funcionales, así como el diagrama de casos de uso.

La planificación temporal del proyecto se puede consultar en el Anexo A.A

3.1 Definición del proyecto

El primer apartado englobará la concreción de este proyecto, clarificando aspectos de las aplicaciones y especificando sus objetivos principales, las funcionalidades desarrolladas y el alcance de este.

3.1.1 Objetivos y funcionalidad

El objetivo de este proyecto es el estudio del *ransomware* en dispositivos móviles Android, lo que comporta su análisis y el estudio de las posibles contramedidas. Para cumplir el objetivo marcado primero analizaremos el comportamiento de diferentes familias de *ransomware* en Android, el funcionamiento y limitación de estas. Utilizaremos por tanto la herramienta HelDroid [7] [22] que nos permite analizar una muestra de manera estática en busca de características propias del *ransomware*. En lo referente a las contramedidas diseñaremos una herramienta de soporte para un administrador de sistemas que permita facilitar la gestión y realización del *backup* de diferentes dispositivos en el seno de una organización.

Para ello crearemos un servidor web programado en Python con el *framework* Flask [23], donde residirá la lógica principal de la aplicación. Los usuarios podrán registrarse desde la web, autenticarse y gestionar sus móviles asociados a su usuario, añadiéndolos o eliminándolos así como listarlos. También podrán ver una tabla con la información de sus *backups*, modificar el *token* de la autenticación en dos pasos por si quieren cambiarlo o han perdido acceso al dispositivo móvil. En el caso de ser un administrador además de las funcionalidades descritas anteriormente podrá obtener un listado de todos los usuarios registrados así como de todos los *backups*. Además el propio servidor enviará de manera autónoma notificaciones al móvil a los usuarios que deban realizar un *backup* porque haya pasado el tiempo prefijado desde su último *backup*.
Y también contaremos con un cliente Android que permita autenticarse, y crear *backups*, subiéndolos al server cuando el usuario desee y eliminándolos.
Ambas aplicaciones contarán con doble factor de autentificación, un usuario y contraseña más un código (*token*). Y las comunicaciones serán a través del protocolo https para garantizar la privacidad y la integridad de los datos transmitidos.

3.1.2 Alcance

Referido a los usuarios del sistema, los clientes estarán formados por los empleados de la empresa u organización, así como otras personas que se considere que también deban tener el dispositivo móvil protegido con *backups* periódicos. Los administradores del sistema, serán los propios administradores de sistemas de comunicaciones de la empresa.
Los móviles serán propiedad de la empresa y tendrán la aplicación ya instalada.

Como se ha indicado previamente, el proyecto creado pretende servir como una parte de un sistema MDM que permita gestionar de forma adecuada los *backups* en un entorno empresarial. No pretende cubrir todas las características que satisfacen los MDM comerciales, ni realizar un análisis dinámico de las aplicaciones Android en busca de *ransomware*.

3.2 Catálogo de requisitos

En esta sección se detallarán los diferentes requisitos funcionales y no funcionales del proyecto.

3.2.1 Requisitos funcionales

A continuación se listarán los requisitos funcionales de la aplicación web en primer lugar, y posteriormente los del cliente Android.

Aplicación Web (servidor Flask)

Subsistema de gestión de cuentas y perfil de usuario

RF 1 Sólo el administrador podrá iniciar el servidor.

RF 2 Un usuario se podrá registrar facilitando su nombre de usuario, su contraseña y escaneando un código QR para la verificación en dos pasos 2SV (*two-step verification*) utilizando protocolo OTP (*one-time password*).

RF 3 Un usuario puede autenticarse e ingresar si previamente se había registrado y los datos que facilita: nombre de usuario, contraseña, y *token* son correctos.

RF 3 Un usuario podrá re-sincronizar su doble factor de autentificación, por si pierde el dispositivo dónde este sincronizado la creación de los *tokens* OTP.

RF 4 Un administrador autenticado podrá listar todos los usuarios de la aplicación y ver cuáles son administradores y cuáles no.

RF 4.1 Podrá dar permisos de administrador a un usuario o revocarle permisos de administrador.

RF 5 Un usuario autenticado podrá salir cerrando su sesión.

Subsistema de gestión de números de móvil

RF 6 Un usuario autenticado podrá registrar un número de móvil. Puede tener registrados varios números, o ninguno. Pero en caso de no tener ninguno no podrá realizar *backups*.

RF 7 Un usuario autenticado podrá eliminar un número de móvil suyo.

RF 8 Un usuario autenticado podrá ver todos los números de móvil asociados a su cuenta.

Subsistema de gestión de *backups*

RF 9 Un usuario autenticado podrá ver una tabla con la información de todos sus *backups*, como el número de móvil, la ruta o rutas del *backup* del móvil, el título (nombre) del *backup*, su descripción, la fecha de creación, la fecha de subida del *backup* en caso de haberse realizado y por último la frecuencia en días para realizar el próximo *backup*.

RF 10 Un administrador podrá listar los *backups* de todos los usuarios y obtener una tabla con la misma información que si un usuario lista sus *backups* además de obtener el nombre de usuario por cada *backup*.

Cliente Android

RF 1 Un usuario podrá autenticarse e ingresar en la aplicación facilitando su nombre de usuario, contraseña, *token* y número de teléfono desde el cual se está autenticando.

RF 2 Una vez autenticado, el usuario podrá ver una lista con información de sus *backups*. Como el título, la fecha de creación, la ruta o rutas del *backup* y la descripción.

RF 2.1 El usuario podrá añadir un *backup* a esa lista facilitando un nombre, una descripción y frecuencia de subida. Después seleccionará la o las rutas de archivos o ficheros o ambos que desee.

RF 3 El usuario podrá seleccionar cualquier *backup* y obtener información más detallada de este como la fecha de subida de ese *backup*, o la frecuencia.

RF 3.1 Podrá eliminar el *backup* seleccionado a través de un botón. Recibirá una respuesta si la acción ha sido realizada satisfactoriamente o no.

RF 3.2 Podrá subir el *backup* a través de un botón. Recibirá una respuesta por cada archivo que suba, detallando si se ha podido subir, si ha habido algún problema y a que puede ser debido el problema: conexión, extensión del archivo u otros.

RF 3.3 Podrá modificar información del *backup* como la descripción y la periodicidad.

RF 4 El usuario podrá acceder a un menú de ayuda con información sobre la aplicación.

RF 5 El usuario podrá acceder a un menú de configuración para establecer sus preferencias, como sólo realizar la subida de *backups* si estás conectado a una red wifi.

RF 6 El usuario podrá salir de la aplicación cerrando sesión.

RF 7 El dispositivo recibirá una notificación cuando el usuario no haya subido un *backup* y hayan pasado los días seleccionados (frecuencia) de ese *backup*. Para que el usuario lo suba al servidor cuando pueda.

3.2.2 Requisitos no funcionales

A continuación, se exponen los requisitos que debe tener el sistema en cuanto a seguridad, eficiencia, usabilidad, mantenimiento, rendimiento y portabilidad.

Seguridad

RNF 1 Las conexiones con el servidor se realizan mediante conexión segura utilizando SSL.

RNF 2 Las contraseñas de los usuarios se almacenan en la base de datos aplicándoles previamente un hash de tipo pbkdf2 Sha1.

RNF 3 El código OTP se crea de manera aleatoria. Y se almacena en la base de datos en base 64.

RNF 4 El código secreto de cada usuario para identificarlos en la aplicación cliente Android se genera de manera aleatoria utilizando la versión 4 del UUID especificado en el RFC 4122 [24], [25].

RNF 5 El servidor y sus componentes deben estar localizados en un entorno físico seguro y protegido por un sistema de seguridad.

RNF 6 Cualquier vulnerabilidad detectada debe ser corregida con el objetivo de asegurar la seguridad e integridad del sistema MDM.

RNF 7 Solo los usuarios registrados podrán acceder a la aplicación.

RNF 8 El sistema mantendrá un log de los eventos que ocurren y las peticiones para poder identificar a los responsables en caso de un mal uso del sistema u obtener el motivo de un fallo en la seguridad del sistema.

RNF 9 El sistema MDM deberá mantener la información almacenada de manera segura para evitar el acceso de personal no autorizado a la misma, garantizando por tanto su integridad.

Eficiencia y rendimiento

RNF 10 La aplicación Cliente en Android deberá ser lo más ligera posible, para economizar el uso de almacenamiento y batería. Además de conseguir una navegación entre menús y opciones de la aplicación fluida. No mayor a 1 segundo en estas transiciones.

RNF 11 El tiempo de acceso al sistema no será mayor a 1 segundo.

RNF 12 El tiempo que tarda el servidor en procesar la petición y devolver la lista de *backups* y el usuario puede verla en el móvil no será superior a 2 segundos.

Usabilidad

RNF 13 Las interfaces de la aplicación para Android siguen el mismo patrón de estilo y diseño, para garantizar una mayor consistencia y familiaridad al usuario. Además de adoptar las pautas de *Material Design* de Google. [26]

RNF 14 La aplicación Android muestra diversos mensajes de validación y error a través de Toast y Snackbars como respuesta a las acciones del usuario.

RNF 15 La aplicación móvil está diseñada para que se adapte a diferentes tamaños de pantalla, densidades y para *tablets*. Utilizando fragmentos.

RNF 16 La aplicación web está diseñada para que sea *responsive* y se adapte a diferentes resoluciones gracias a Bootstrap.

RNF 17 La aplicación será desarrollada en inglés al igual que la interfaz de la misma. Para facilitar su integración en un entorno trabajo multipaís.

Mantenibilidad

RNF 18 El sistema será escalable, siendo fácilmente extensible tanto en número de usuarios, como en funcionalidades, ya que se construirá el sistema para que permita modificar o añadir nuevos tipos de funcionalidad, sumándose a los ya ofrecidos por el sistema.

RNF 19 La aplicación móvil sigue una estructura modular, por lo que su código se encuentra divido en diferentes paquetes según su funcionalidad. Separando las actividades, el modelo, el servidor y Firebase.

RNF 20 Tanto el servidor como la aplicación móvil están comentados para facilitar su mantenimiento y sus futuras ampliaciones.

Portabilidad

RNF 21 El usuario podrá acceder al servicio web independientemente de la localización, mientras que disponga de su teléfono o generador de *tokens* y conexión a Internet.

Sistema

RNF 22 El cliente Android deberá funcionar en dispositivos con una API de versión 16 o superior

3.3 Casos de uso

En esta sección de presentarán los casos de uso del proyecto desarrollado, realizando una distinción entre la aplicación web conjuntamente con el servidor y el Cliente Android. Los casos de uso aquí representados se han obtenido a partir de los requisitos funcionales especificados para el sistema.

A continuación se puede observar el diagrama de casos de uso para la aplicación web y el servidor, donde se muestran representadas las posibles acciones mencionadas en el catálogo de requisitos, para cada uno de los actores que formar parte del proyecto: Los usuarios y los administradores (ver Figura 6).

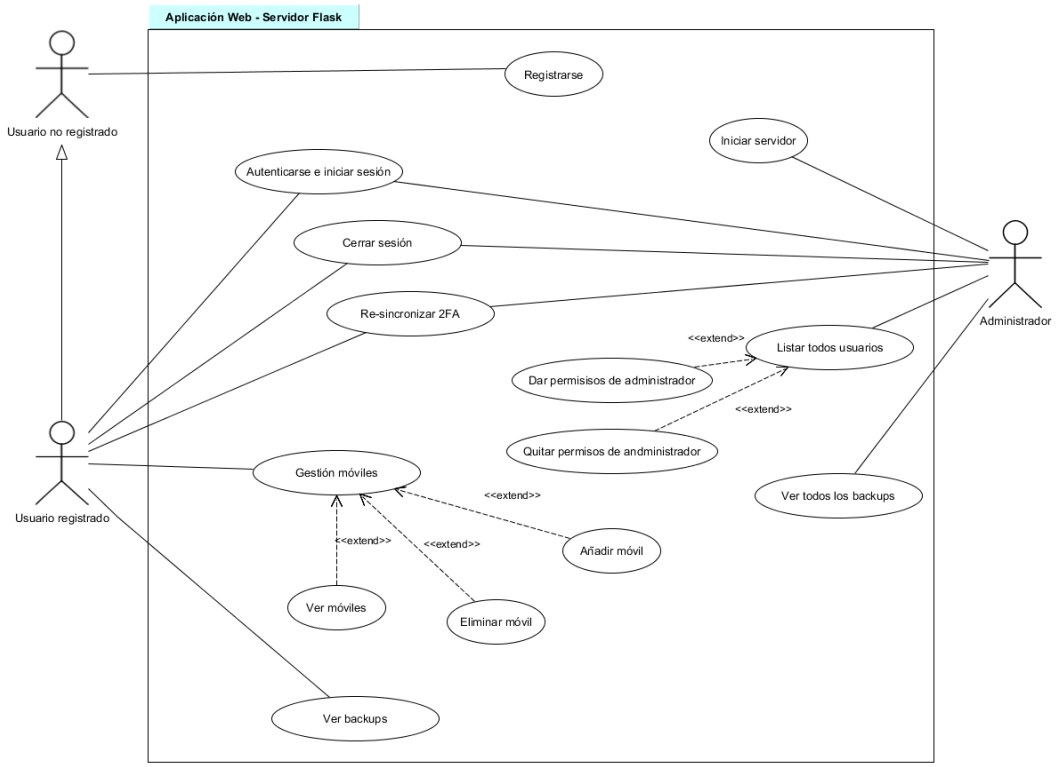

Figura 6: Diagrama de casos de uso de la aplicación web – Servidor Flask.

El siguiente diagrama será el de casos de uso para el cliente Android (ver Figura 7). En este caso solo hay un actor el usuario registrado. Pero todas las acciones excepto ayuda y configuración están conectadas con el servidor. Se ha obviado en el diagrama para facilitar su lectura.

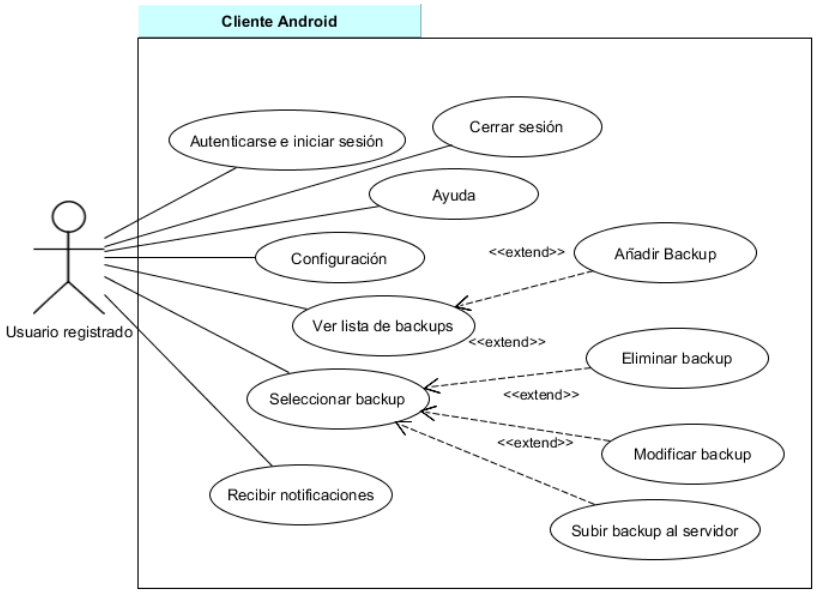

Figura 7: Diagrama de casos de uso del cliente Android.

A continuación, se procede a detallar los dos casos de uso más representativos del sistema.

3.3.1 CU01: Añadir un backup

Nombre Caso de Uso: Añadir un *backup.*
Actores primarios: Usuario del sistema (está registrado y ha iniciado sesión) conectado a través de la aplicación cliente Android.
Interesados y objetivos:

- **Usuario del sistema:** Quiere añadir un nuevo *backup* en el sistema para posteriormente poderlo subirlo al servidor.

Resumen: El usuario del sistema ingresará en la aplicación cliente Android y seleccionará añadir *backup*. Introducirá su título, descripción, frecuencia y la ruta de los archivos y/o carpetas. El nuevo *backup* quedará registrado en el sistema.
Precondiciones: El usuario del sistema se ha autenticado como tal.
Garantía de éxito (post-condiciones): El nuevo *backup* queda registrado en el sistema y el usuario podrá modificarlo, eliminarlo o subirlo al servidor.
Escenario principal de éxito:

1. El usuario del sistema selecciona la opción de añadir un nuevo *backup.*
2. El sistema presenta en la pantalla del dispositivo Android los cuadros de texto para introducir el título, descripción y frecuencia.
3. El usuario introduce el título.
4. El usuario introduce la descripción.
5. El usuario introduce la frecuencia.
6. El usuario confirma la introducción.
7. El sistema presenta un diálogo para seleccionar los archivos y/o carpetas que conformarán la ruta.
8. El usuario elige la ruta.
9. El usuario confirma la introducción.
10. La aplicación móvil envía el *backup* al servidor quien lo almacena, y por lo tanto queda registrado en el sistema. Y la aplicación muestra en pantalla la lista de *backups* del usuario.

Extensiones (flujos alternativos):

6. El usuario confirma la acción pero el título o descripción tienen una longitud menor a tres caracteres. O la frecuencia es un número negativo o está vacío.
    a. El sistema le muestra un mensaje de error explicando dónde está el fallo y se cancela la operación.
1. El usuario confirma la introducción pero no ha seleccionado ninguna ruta
    a. El sistema le muestra un mensaje de error explicando que debe seleccionar una ruta válida y se cancela la operación.

3.3.2 CU02: Subir un *backup* al servidor

Nombre Caso de Uso: Subir un *backup* al servidor.
Actores primarios: Usuario del sistema (está registrado y ha iniciado sesión) conectado a través de la aplicación cliente Android.
Interesados y objetivos:

- **Usuario del sistema:** Quiere subir el *backup* seleccionado al servidor para poder tener una copia de seguridad de ese *backup*.

Resumen: El usuario del sistema ingresará en la aplicación cliente Android y seleccionará el *backup* que desee de la lista de sus *backups*. Una vez seleccionado, pulsará el botón de

subir *backup* (upload). Conforme se vayan subiendo los archivos se le mostrará un mensaje del progreso.

Precondiciones: El usuario del sistema se ha autenticado como tal.

Garantía de éxito (post-condiciones): El *backup* ha sido subido con éxito al servidor y todos los ficheros pertinentes han sido guardados en el servidor. Al ser un *backup* incremental, en subidas sucesivas solo se suben las modificaciones o nuevos ficheros.

Escenario principal de éxito:

1. El usuario del sistema selecciona un *backup* de la lista de sus *backups.*
2. El usuario pulsa el botón para subirlo al servidor.
3. Si es la primera vez que sube ese *backup*:
   3.1 Los archivos se suben al servidor.
   3.2 Se muestra un mensaje al usuario confirmando que el fichero ha sido subido.
4. Si no es la primera vez que sube ese *backup*:
   4.1 Los archivos nuevos o modificados desde la última vez que se subió el *backup* se suben al servidor.
   5.2 Se muestra un mensaje al usuario confirmando que el fichero ha sido subido.

Extensiones (flujos alternativos):

3 y 4: El archivo contiene una extensión no aceptada por el servidor.

a. El archivo no se guarda en el servidor y se notifica al usuario que la extensión no es válida.

3 y 4: Se pierde la conexión con el servidor durante la subida de un archivo.

a. El archivo no se guarda en el servidor y se notifica al usuario que la subida del archivo no ha podido ser completada.
b. Cuando se restablezca la conexión se reintentará la subida, y en caso satisfactorio se notificará al usuario que se ha completado la subida.

4 Diseño

En esta sección se especificará el diseño del cliente web y la aplicación cliente Android, el flujo de navegación del sistema y la estructura de la aplicación y de la base de datos, para cumplir con los requisitos detallados en el capítulo anterior.

Se hará especial hincapié en la explicación del porqué de las decisiones tomadas justificándolas de manera lógica.

4.1 Definición del diseño

Nuestro diseño, tanto de la aplicación web como del cliente, han tenido como objetivos facilitar la escalabilidad del sistema y reutilización del código. Eso se ha podido lograr siguiendo una estructura modular, separando en paquetes, archivos y funciones las aplicaciones. Estas decisiones nos permiten tener la máxima cohesión y el mínimo acoplamiento posible. Además de facilitar el mantenimiento de las aplicaciones.

La funcionalidad principal de las aplicaciones web o móvil es la correcta gestión de los *backups* de un usuario, es por ello que el diseño ha sido concebido con unas características muy concretas, en base a unas especificaciones para que los usuarios puedan sentirse cómodos y maximizar el uso en la herramienta sin ver deteriorada su experiencia personal.

Estas son las especificaciones que deben cumplir ambas interfaces:

- **Estructurada:** La interfaz debe presentar una organización adecuada en los diferentes entornos.

- **Consistente:** Los elementos de diseño deben seguir una línea pareja visual y de diseño, facilitando la comodidad del usuario con la interfaz. En la web esto lo logramos utilizando Boostrap, unas de las tecnologías líder en el sector utilizadas en multitud de páginas. Aparte de un formato con plantillas para que todas las páginas sean similares. En la aplicación cliente Android se ha logrado gracias al uso de Material Design de Google, siguiendo sus líneas de diseño.

- **Tiempo de respuesta y procesado:** El tiempo de respuesta (presentar la información en la pantalla del dispositivo móvil o del PC) debe ser lo más bajo posible, en particular menor a 1 segundo para la mayoría de acciones y menor a 2 segundos para cargar la lista de *backups* en el móvil.

- **Informativa:** Se muestran mensajes de información del resultado de las acciones realizadas. Notificando al usuario en caso de error, e informando del motivo. En relación a la consistencia descrita anteriormente, los mensajes de error de la aplicación web y móvil son idénticos para facilitar la comprensión al usuario.

- **Sencillez e intuitiva:** El diseño tiene que seguir unas pautas lógicas y que el usuario pueda realizar las acciones de manera directa sin tener que pararse a buscar. Es por ello que debe ser sencilla, facilitando el uso de las herramientas a diferentes usuarios sin importar su nivel de conocimientos. Tampoco se debe abrumar al usuario con demasiadas acciones o mensajes muy largos. Así como no necesitar múltiples pasos para realizar una acción determinada.

4.2 Arquitectura del sistema y flujo de operación

El sistema está formado por un cliente Android MDM y un servidor MDM con el que se puede interaccionar a través de una aplicación web. De esta manera los usuarios podrán editar información y gestionar sus números de móvil y sus *backups* desde la página web, y desde el móvil podrán subir los *backups* al servidor y ser notificados cuando deban subir el próximo *backup*.

A continuación vamos a explicar e ilustrar uno de los flujos principales del proyecto dónde un usuario se registra y sube un *backup* al servidor. Para que podamos conocer mejor la presente arquitectura del proyecto (ver Figura 8).

Lo primero que hace el usuario es registrarse en el sistema a través del interfaz web, en caso de registro correcto guardará los datos pertinentes en la base de datos (nombre de usuario, el hash de su contraseña, el tipo de usuario que por defecto es usuario normal, así como el secreto OTP para la 2SV). En el registro el usuario deberá escanear el código QR con un móvil- por ejemplo- con alguna aplicación de 2SV o 2FA instalada como puede ser Google Authenticator [27] o FreeOTP Authenticator [28] como alternativa de código abierto.
Posteriormente deberá autenticarse con el usuario, contraseña y código que le facilitará la aplicación 2FA. Una vez autentificado en su página de perfil (user Page) podrá añadir un número de teléfono móvil.

Ahora solo debería autenticarse desde la aplicación cliente facilitando su usuario, contraseña, código de 2FA y el número de móvil desde el que se está conectado. Estos datos son enviados cifrados a través de SSL al servidor conjuntamente con el ID de Firebase que la propia aplicación ha obtenido al instalarse por primera vez. Este ID de Firebase es lo que permite identificar un móvil de manera unívoca para enviarle las notificaciones desde el servidor.
El servidor, en caso de autenticación satisfactoria devolverá al cliente un identificador único de usuario generado aleatoriamente. Este identificador al identificar un usuario permite que no deba volver a iniciar sesión en la aplicación a menos que cierre su sesión. Eso es debido a que el servidor ha almacenado ese identificador asociado al usuario y por lo tanto puede verificar la autenticidad de la conexión.

Dentro de la aplicación móvil podrá añadir un *backup* introduciendo su título, descripción, frecuencia con la que debe realizarse y seleccionado una o varias rutas de archivos y/o carpetas a almacenar. Una vez creado, le aparecerá en la lista de *backups* del usuario, dónde podrá seleccionar uno de ellos. Después de seleccionarlo, en la pantalla habrán 3 botones, para editar información del *backup* (edit), para subirlo (upload) y para eliminarlo (remove). Para subirlo simplemente presionar el botón de *upload* y la propia aplicación le irá notificando al usuario conforme cada archivo se suba al servidor, avisándole en caso de fallo y explicándole el motivo.

En el diagrama el usuario está representado como  , la aplicación web  , el servidor  , el cliente Android  y por la app 2FA que genera los OTP .

Las flechas denotan el sentido de la comunicación. Se han omitido las flechas de *return* Ok y Error que devuelve toda comunicación con el servidor y sus mensajes para mejorar la claridad del esquema.

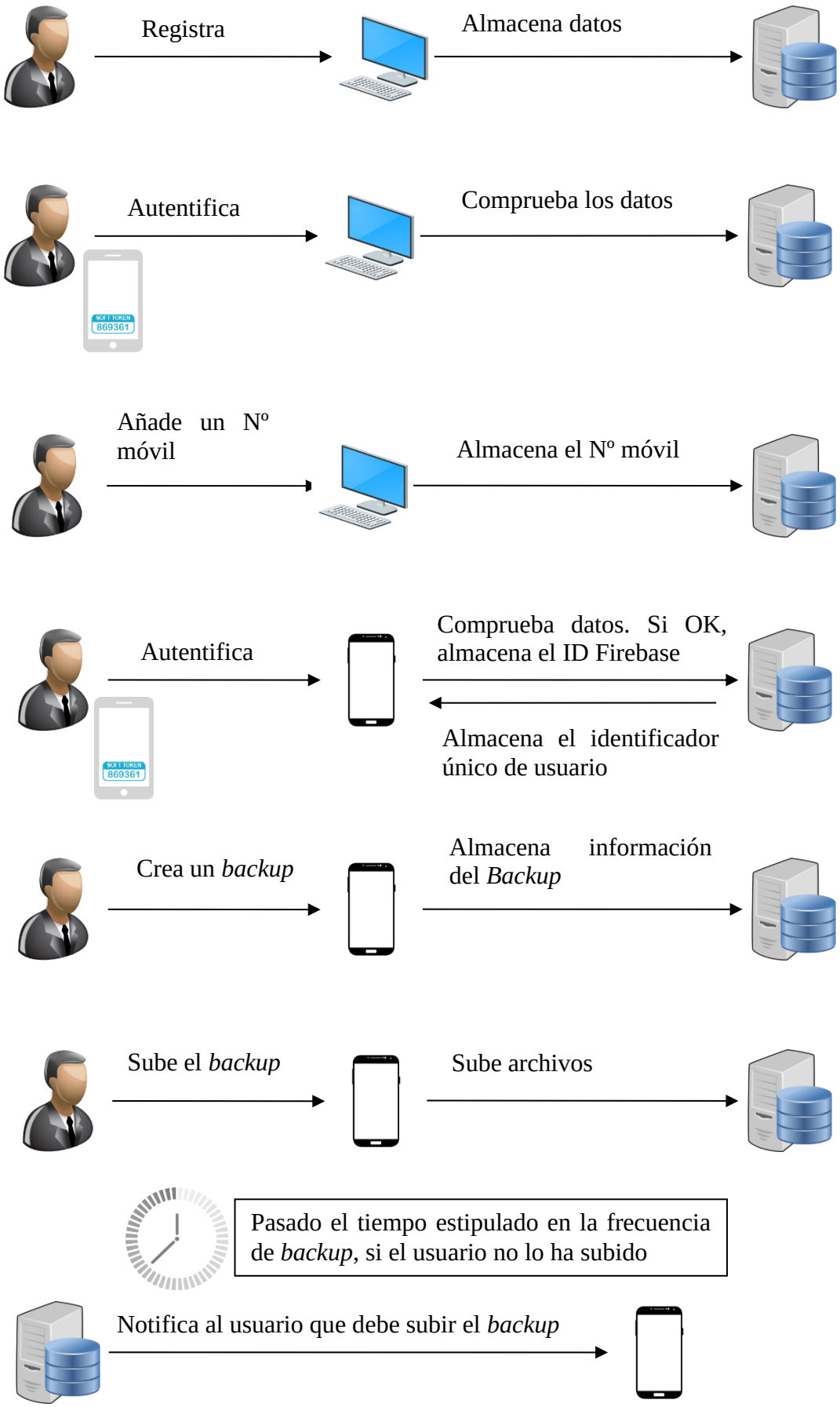

Figura 8: Flujo de operación para un registro y subida de un *backup*.

4.2.1 Decisiones en el diseño

En este apartado vamos a explicar el motivo de ciertas decisiones de diseño. Las decisiones relacionadas con el software, herramientas y *framework* empleado se explicarán más adelante en el apartado 5.2 Herramientas empleadas.

No se puede registrar un usuario desde el cliente Android:
En la aplicación los usuarios solo se podrán registrar desde la aplicación web ya que en muchas ocasiones la aplicación 2FA que el usuario utilizará la tiene instalada en el mismo dispositivo que la aplicación cliente de Android. Y resultaría más complicado el proceso de registro ya que debería escanear un código QR que se está mostrando en la pantalla del dispositivo con la propia cámara del dispositivo.

Extensiones aceptadas para los ficheros del *backup*:
Los archivos de *backup* que puede subir un usuario están limitados a una lista de extensiones. En particular a las extensiones que suelen cifrar los ransomwares y por tanto son los archivos que debemos tener en *backup*. Fijándonos en la lista del ransomware Alpha Crypt [29], y se ha añadido alguna extensión adicional como .gif, .mp3, .flac, .cvs, ya que este tipo de ficheros podrían contener información valiosa para el usuario (e.g. notas de una reunión en formato audio).
Aunque Alpha Crypt sea un *ransomware* que cifra archivos en Windows, las extensiones son también usadas en el móvil Android porque son ficheros que suelen provenir de su estación de trabajo.

Tamaño máximo del archivo de *backup*
Actualmente no hay limitación de tamaño para el archivo a subir de *backup* ya que se entiende que todos los archivos que se suban son de importancia para la empresa. No obstante, si se detectara un problema con el tamaño de los *backups* se podría poner un máximo.

***Backup* incremental**
Se ha optado por una estrategia de *backup* incremental, es decir, en los *backups* sucesivos solo se copian al servidor los nuevos archivos o las modificaciones. Para poder aprovechar al máximo el espacio en el servidor. Esto se realiza comprobando si la fecha de modificación del archivo a subir es mayor a la fecha de última subida al servidor del *backup*. Ambas fechas son insertadas en el lado del cliente Android, para evitar problemas de sincronización horaria.

Utilización de 2FA
La aplicación se ha diseñado con miras en la seguridad informática y en poder recuperar los archivos del teléfono de un usuario. Es por ello que creemos debemos evitar que un cibercriminal robe los datos a un usuario, por lo que se ha optado por implementar doble factor de autentificación mediante contraseñas de un solo uso [30]. Se ha optado por este esquema (una aplicación) en vez de recibir un mensaje SMS dados los problemas que presentan esta última alternativa como el *roaming* internacional, el coste del mensaje, los retrasos, la seguridad y en conclusión, la poca certeza sobre su viabilidad [31].

SSL
Para garantizar la seguridad e integridad de los datos que se transmitan entre las aplicaciones (web y Cliente Android) con el servidor implementaremos cifrado SSL.

4.3 Diseño del Servidor y la aplicación Web

En este apartado se explicará el diseño de la aplicación web desde el punto de vista del aspecto lógica y gráfico. Además, se detallará la estructura de la base de datos del servidor. El servidor es el elemento que realiza el grueso de las operaciones.

4.3.1 Base de datos del Servidor

La base de datos que almacena toda la información del servidor está compuesta por tres tablas. Como se puede comprobar en la Figura 9, se trata de una base de datos muy sencilla cuya finalidad es principalmente almacenar la información pertinente a los *backups* de un usuario, y en la tabla phones se almacena el REGID, que es el código que permite mandar notificaciones a ese dispositivo a través de Firebase. Y en la tabla de users, destacar los campos Otp_secret y Personal_code, que son el secreto para la 2FA y el código secreto único a cada usuario respectivamente. Tipo_usuario indica si se trata de un administrador o de un usuario corriente de la aplicación.

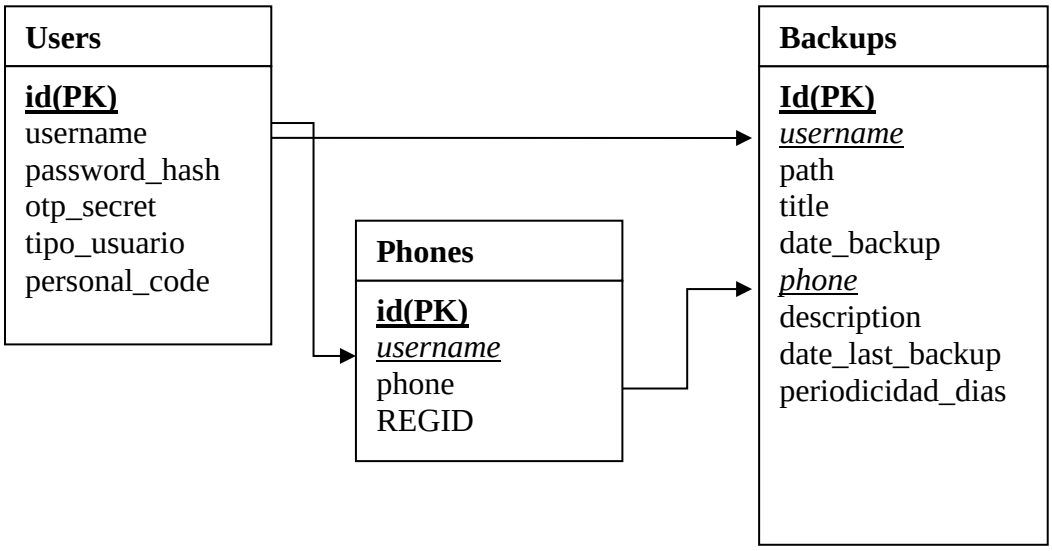

Figura 9: Diagrama modelo-relacional.

Las claves primarias se representan en negrita y subrayado y (PK). Las claves foráneas se representan en cursiva y subrayado, con una flecha indicando la relación.
Además de estas restricciones de unicidad y de referencia, se van a establecer las siguientes restricciones:

- **Users:** todos los campos serán NOT_NULL (no podrán ser vacíos) a excepción de personal_code que estará vacío hasta que el usuario se autentique desde la aplicación móvil. El campo username será UNIQUE, para que no existan nombres de usuario repetidos.
- **Phones:** el campo phone será UNIQUE. Los campos username y phone serán NOT_NULL. Por último, el campo REGID estará vacío hasta que se obtenga el código regid del dispositivo (al autenticarse desde la aplicación móvil)
- **Backups:** todos los campos serán NOT_NULL excepto date_last_backup que estará vacío hasta que no se suba el backup al servidor. El campo title o description no pueden ser UNIQUE porque es una tabla de *backups* para todos los usuarios, y diferentes usuarios pueden poner el mismo título. El campo periodicidad_dias debe ser un entero positivo.

4.3.2 Diseño lógico

El servidor web ha sido desarrollado en Python con el *framework* Flask por lo tanto las páginas webs han sido desarrolladas como .html que extienden plantillas Jinja2. La explicación de las tecnologías empleadas será descrita en el apartado 5.2 Herramientas empleadas.

Una vez especificado el modelo y arquitectura del proyecto, se va a proceder a especificar el mapa de navegación de la aplicación web, definiendo las pantallas y los flujos de navegación posible entre ellas. Como se puede observar en la Figura 10.

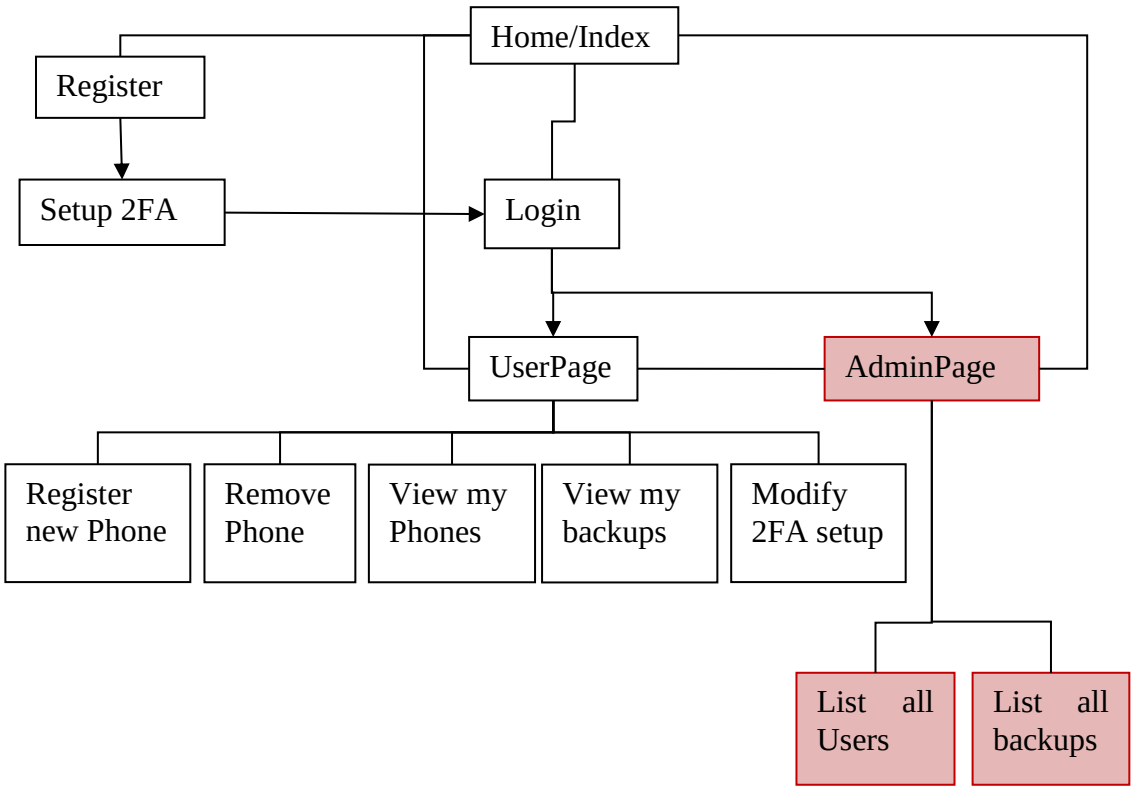

Figura 10: Mapa de navegación de la aplicación web.

Cada pantalla del mapa de navegación se corresponde con un fichero .html. Destacar que en cualquier página el usuario tiene un enlace "Log out" para cerrar sesión en la aplicación.

Los recuadros rojos son solo accesibles y visibles para los administradores del sistema.

4.3.3 Diseño gráfico

En lo referente al diseño gráfico de la aplicación web, se utilizará el mismo estilo en todas las páginas, haciendo uso de Bootstrap [32] para que se pueda adaptar a los diferentes tamaños de pantalla. Y se buscará presentar la información de manera clara y concisa. A continuación se muestran dos ejemplos de las pantallas principales de la aplicación web: la pantalla de acciones que puede realizar un usuario (ver Figura 11), y la pantalla que aparece tras pulsar "View my backups" y que incluye una tabla con información de los *backups* de un usuario (ver Figura 12).

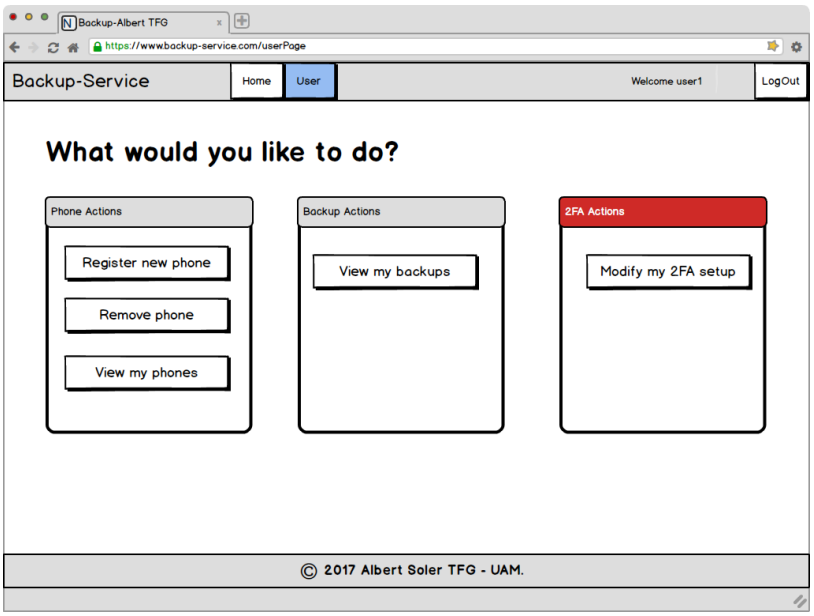

Figura 11: Maqueta web - Página de usuario.

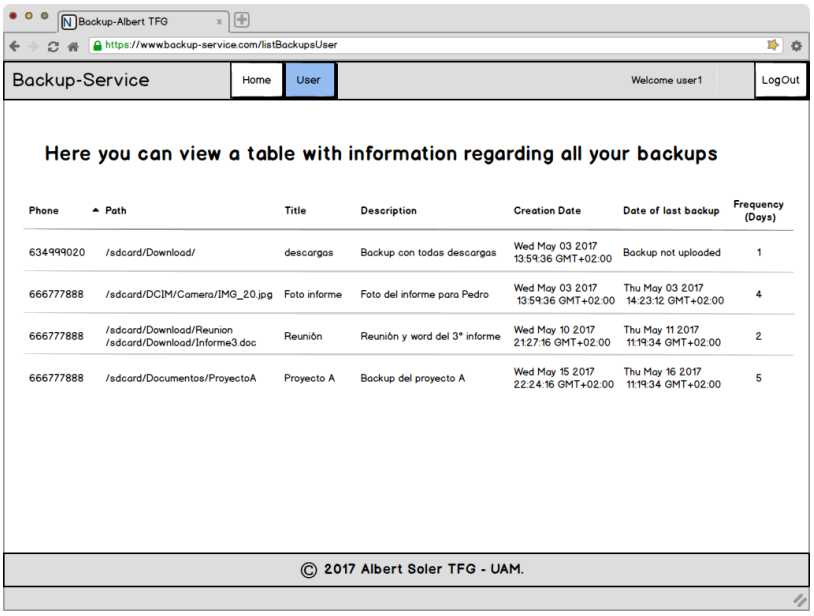

Figura 12: Maqueta web - Ver todos los *backups* de un usuario.

4.4 Diseño del cliente Android

Terminado el diseño relativo al servidor, nos centraremos en el diseño de la aplicación cliente de Android, desde el aspecto lógico y gráfico.

4.4.1 Diseño lógico

El diseño de la aplicación Android se centra en responder a las necesidades del usuario vistas en la sección de análisis (ver Sección 3). En la figura siguiente se pueden observar el diagrama con las actividades de la aplicación, su flujo de interacción y las principales funcionalidades (ver Figura 13).

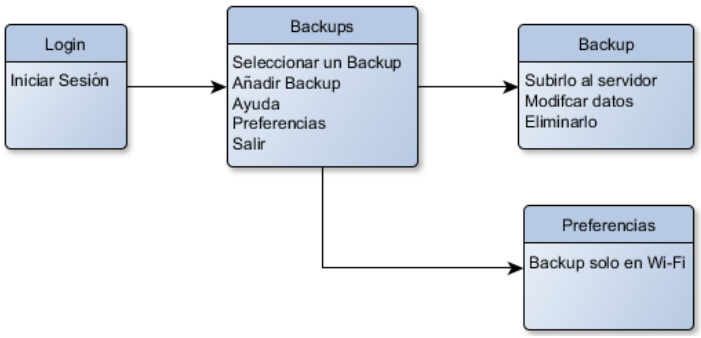

Figura 13: Diagrama de las actividades de la aplicación Android.

Como se puede observar es necesario iniciar sesión (identificarse y autenticarse) para continuar usando la aplicación lo cual es lógico ya que es un requisito para podernos conectar con el servidor.
La aplicación también mostrará las notificaciones Firebase que reciba del servidor como mensajes *push* de Android, para que el usuario sea consciente de la necesidad de subir un *backup* al servidor a pesar de tener la aplicación cerrada.
La actividad principal será la lista de *backups* (abreviada *Backups* en el diagrama) dónde podremos seleccionar uno de ellos para subirlo al servidor (entre otras acciones como son modificar sus datos y eliminarlo).

4.4.2 Diseño gráfico

La parte gráfica de la aplicación cliente de Android ha sido desarrollada teniendo en mente los principios de usabilidad, sencillez de uso, que se adapte a diferentes tamaños de pantalla y que sea lo más parecida posible a interfaces que sean familiares para el usuario. Consiguiendo con ello que sea una aplicación amigable con un uso muy intuitivo.
A continuación, mostraremos un par de maquetas de las actividades más importantes de la aplicación Android. La actividad principal es una lista de *backups* del usuario para ese móvil (ver Figura 14). En la Figura 15 podemos ver la pantalla cuando seleccionamos un *backup* y lo subimos al servidor. Se puede observar el mensaje de acción realizada de manera satisfactoria que obtenemos por cada elemento subido, en caso contrario recibiríamos un mensaje explicando el fallo ocurrido.

Figura 14: Maqueta Android – Lista de *Backups*.

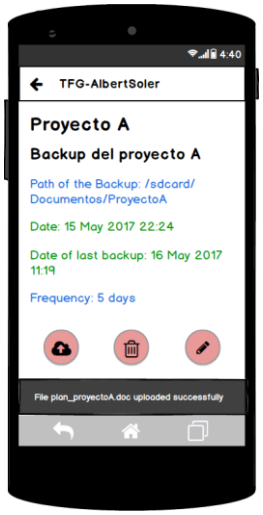

Figura 15: Maqueta Android – Subir un *backup* al servidor.

4.5 Requisitos técnicos

Para que todos los elementos (servidor, aplicación Web y aplicación Android) funcionen de manera correcta deberán de cumplirse unos requisitos mínimos.

4.5.1 Servidor Rest

- **Python 2.7** Para el desarrollo del código del Servidor se ha utilizado la versión 2.7 de Python aunque también podría utilizarse la 3.6 que es compatible con Flask. Sin embargo, existen funciones no retro-compatibles y menor número de librerías disponibles, por ello se ha decidido utilizar la versión 2.7 [33]
- **Flask** Un *framework* ligero para crear un servidor Python [23]
- **Conexión a internet:** Al tratarse de un servidor necesita conexión para recibir peticiones REST y para enviar notificaciones Firebase.
- **Pip:** Herramienta que facilita la instalación de todos los paquetes necesarios de Python para que el servidor pueda funcionar.[34] Como por ejemplo: pyOpenSSL para garantizar la seguridad en las comunicaciones, Flask-RESTful para facilitar crear un servidor de tipo REST, Flask-SQLAlchemy para comunicarse y operar con la base de datos SQLite.
- **Base de datos:** No es necesario ya que el propio código Python crea una base de datos SQLite auto-contenida. [35]

4.5.2 Aplicación Web

- **Conexión a Internet:** Como es lógico se trata de una aplicación web por lo que se necesita una conexión a internet para poder funcionar y enviar peticiones al servidor.
- **Un navegador web:** Misma razón que la anterior, necesitaremos navegar por diferentes páginas y visualizar la información. No es necesario tener la última versión del navegador pero si recomendable, principalmente por la seguridad. Ya que un navegador antiguo sin parches y actualizaciones de seguridad puede convertirse en una puerta de entrada para robar datos de los usuarios.

4.5.3 Aplicación Android

Al igual que con la aplicación Web, apenas hay requisitos.

- Sistema operativo **Android** con una versión igual o superior a la **API 16(**Jelly Bean)
- **Conexión a internet:** Misma razón que para la aplicación web además de la necesidad de recibir notificaciones del servidor. La conexión puede ser tanto wifi como a través de datos móviles. Pero realizar el *backup* por datos puede suponer un sobrecoste en la factura del móvil.

5 Desarrollo

El propósito de este capítulo es el de listar todo lo relacionado con la implementación del proyecto. Primero se comenzará por cómo se ha implementado las diferentes partes, posteriormente se explicarán las herramientas y lenguajes utilizados en el desarrollo así como las características hardware y software del equipo de desarrollo.
A continuación se comentará la estructura del servidor y aplicación web, listando los ficheros que lo componen y explicando brevemente su funcionalidad. Se proseguirá, de manera análoga, con la explicación del cliente Android, detallando sus paquetes y clases.

5.1 Implementación

Una vez determinados los requisitos y el diseño del proyecto, se ha procedido a realizar el desarrollo del mismo. Para ello se ha seguido una metodología basada en prototipos funcionales, sobre los cuales se han ido validando los requisitos, añadiendo o modificando en caso necesario.

Los primeros prototipos del cliente Android y Web son las maquetas mostradas en el capítulo de Diseño. Posteriormente se realizó la aplicación web probando por tanto la funcionalidad del servidor, se incorporó SSL y la autentificación en dos pasos para permitir el registro y autenticación de usuarios. En la siguiente etapa se desarrolló la aplicación Android pero solo el apartado del Firebase el cual se comprobó a través de la consola de Firebase de Google. Mientras tanto también se codificó las diferentes tablas de la base de datos y las páginas de usuario y administrador.
Lo siguiente fue avanzar en la aplicación Android para poder crear *backups*, es decir, la lógica de conexión con el servidor REST. Mientras en el servidor se introdujo el envío de notificaciones al móvil. Y después se añadió la lógica necesaria para poder subir *backups* al servidor.

Este proceso de prototipos incrementales se ha ido repitiendo hasta la obtención de un prototipo final, sobre el cual no se ha visto la necesidad de realizar ninguna modificación o mejora por lo que se ha terminado el proceso. Este prototipo final consiste en el Servidor, aplicación cliente Android y aplicación Web.

En el anexo B, Manual de instalación, se encuentra una guía para poder instalar el proyecto. Así como crear los certificados necesarios para poder utilizar SSL.

5.2 Herramientas empleadas

Para realizar este proyecto se han utilizado diversas herramientas y lenguajes de programación que detallaremos a continuación.

5.2.1 Flask

Flask es un micro *framework* ligero codificado en Python. Permite codificar un servidor de manera rápida y sencilla, también es usado por grandes compañías como por ejemplo Pinterest [36].
Tiene la ventaja de poder utilizar los diferentes paquetes y bibliotecas para Python aparte de los suyos propios, extendiendo así su funcionalidad.

En el proyecto se han utilizado bastantes paquetes adicionales como: **pyfcm** para enviar las notificaciones Firebase al cliente Android; **OpenSSL** para realizar las conexiones a través de SSL y **flask_sslify** para forzar que todas las conexiones sean a través de SSL; **Flask_restful** para la Api REST; **Wtforms** para generar formularios y validarlos; **uuid** para la generación de identificadores únicos; **onetimepass** para validar el Token (contraseña de un solo uso) y **pyqrcode** para generarlo; **apscheduler** para crear un planificador de *backups*.
Además, se utilizan los paquetes y extensiones más comunes de Flask para el registro, *login*, manejo de sesiones y gestión de la base de datos SQLite. (**werkzeug.security**, **flask.ext.login** y **flask.ext.sqlalchemy**)

Los motivos por los cuales se ha decidido utilizar Flask en este proyecto han sido por las características aquí descritas, principalmente la facilidad para crear un servidor y comenzar a probarlo y poder ajustarlo y extenderlo a las necesidades concretas. Además, al no haber trabajado previamente con Flask ha servido también como labor de aprendizaje de un nuevo *framework* muy utilizado actualmente.

5.2.2 Python

Al haber seleccionado Flask como framework para el servidor era necesario seleccionar Python como lenguaje principal de programación del servidor y la aplicación web. Se trata de un lenguaje de programación de alto nivel de propósito general. Conocido por ser fácilmente "leído" y "entendido" así como utilizar los espacios en blanco (la *indentación* o sangrado) para delimitar bloques de código. Y como se ha mencionado en la subsección anterior, se han utilizado varias de sus múltiples bibliotecas para extender su utilidad.

5.2.3 OpenSSL

Aparte de ser utilizada en el servidor, también ha sido utilizada para la creación de los certificados auto-firmados para el servidor y cliente. Certificados que han sido utilizados para la comunicación segura a través de SSL entre los diversos componentes.

5.2.4 Android

Como es lógico se ha utilizado Android SDK con el lenguaje de programación Java para desarrollar la aplicación para Android, teniendo en cuenta las guías de estilo de Material Design para la interfaz gráfica.

Para la conectividad con la API Rest del Servidor se ha utilizado la librería HTTP Volley [37] y VolleyMultipartRequest [38] para la subida de archivos al servidor.

Para poder recibir las notificaciones en el móvil, se ha utilizado la herramienta FCM (Firebase Cloud Messaging) [39]. Permite a través de la Consola de notificación de Google o de nuestro servidor Flask enviar notificaciones de manera gratuita a los clientes que tengan nuestra aplicación Android instalada gracias al ID único de cada dispositivo. El envío de notificaciones se realizará desde un planificador que cada 6 horas comprobará si hay usuarios a los cuales notificar que deben subir un *backup* al servidor en función de la fecha de última subida de ese *backup* y la frecuencia de subida dada.

5.3 Equipo de desarrollo

A continuación se listan las características del equipo en el cual se ha desarrollado el proyecto. Primero las características Hardware y a continuación el Software.

5.3.1 Hardware

El proyecto ha sido llevado a cabo en un Dell XPS 15Z con las siguientes características:

- Procesador Intel Core i7 2620M
- Tarjeta gráfica dedicada NVIDIA GeForce GT 525
- Memoria RAM 8 Gb DDR3

5.3.2 Software

Sistemas operativos:

- **Windows 10 Home 64 bits** El Sistema operativo principal del equipo, donde se han desarrollado todas las aplicaciones y el servidor. Así como donde se ejecuta el servidor.
- **GenyMotion** Emulador de Android para realizar pruebas, sobre todo relacionadas con el ransomware para no infectar un teléfono móvil físico [40].
- **Android 6.01 Nexus 5**, para las pruebas de la aplicación cliente

Software de programación

- **Sublime Text 3** para la codificación del servidor.
- **Android Studio 2.2.3** Para la programación de la aplicación cliente Android [41].
- **Git** y **BitBucket** para la gestión de versiones y copias de seguridad.
- **DB Browser for SQLite** para la base de datos [42].

Software de edición

- **Microsoft Word** Para la redacción del presente documento y conversión a PDF.
- **Balsamiq Mockups** Para la creación de las maquetas de la web y del servidor [43]**.**
- **yED** Para la creación de diagramas [44].

Software de pruebas

- **Postman** Para probar el correcto funcionamiento de la API REST diseñada [45].
- **Códigos QR:** Se ha utilizado dos aplicaciones para gestionarlos
    - **Google Authenticator** [27]: una aplicación presente en muchos teléfonos móviles.
    - **FreeOTP Authenticator** [28]: una alternativa de código abierto.

Bibliografía y referencias

- **Mendeley** Una aplicación que permite gestionar referencias y fuentes. Además, Mendeley dispone de un *plug-in* para Microsoft Word que permite insertar las referencias y la bibliografía [46].

5.4 Estructura y documentación del código

El código desarrollado para el cliente Android, la aplicación Web y servidor se ha modulado atendiendo a su funcionalidad. A continuación, se describirán los paquetes y clases del cliente Android, y los ficheros de la aplicación Web y del servidor.

5.4.1 Servidor y aplicación web

Los ficheros plantilla para crear las páginas web (.html) se encuentran en una carpeta llamada: *templates*. Los ficheros de hoja de estilo en cascada (.css) se encuentran en *static/css*. Los certificados están en la carpeta *resources*. En la carpeta *backups* se encuentran todos los *backups* de los usuarios de la aplicación.

- **config.py:** fichero de configuración del server. Indica la ruta de las base de datos.
- **requirements.txt:** fichero que incluye las dependencias necesarias de Flask y Python. Facilita la instalación de estas ejecutando el comando:
  pip install –r requeriments.txt
- **README.md:** contiene un pequeño "léeme" con información del proyecto, autor y manual de instalación.
- **LICENSE:** contiene la licencia y Copyright del proyecto.
- **app.py:** fichero principal con toda la lógica del servidor y de la aplicación web. Incluyendo la API REST.
- **base.html:** plantilla base de la cual heredan el resto de páginas, contiene los menús y el pie de página.
- **index.html:** página principal con los links para autenticarse o registrarse. Contiene un párrafo de información del proyecto.
- **register.html:** página de registro con un formulario para introducir el nombre de usuario y la contraseña. Añade el nombre de usuario, el hash de la contraseña y un secreto para 2FA (que crea de manera aleatoria) a la tabla **users** de la base de datos.
- **two-factor-setup.html:** página a la cual se accede después del registro para configurar la autenticación en dos pasos. Incluye el código QR que hay que escanear. El código QR se forma gracias al secreto 2FA creado anteriormente.
- **login.html:** página para autenticarse, incluye un formulario para introducir nombre de usuario, contraseña y *token* de 2FA. Comprueba que los tres campos son correctos, en caso contrario muestra un mensaje de error diciendo que alguno de ellos es incorrecto (para no dar información ante un posible ataque).
- **userPage.html:** página de las acciones que puede hacer un usuario autenticado, desde aquí se acceden a register-phone-user.html, remove-phone-user.html y list-phone-user.html, view-backups-user.html y change-two-factor-setup.html.
- **register-phone-user.html:** página para registrar un número de teléfono asociándolo al usuario. Incluye un formulario para introducir el número.
- **remove-phone-user.html:** página para eliminar un número de teléfono asociado al usuario. Incluye un formulario para introducir el número.
- **list-phone-user.html:** página que permite ver a un usuario sus números de teléfono asociados.
- **view-backups-user.html:** página que permite visualizar una tabla con la información de los *backups* del usuario.
- **change-two-factor-setup.html:** página que permite cambiar la secreto de 2FA de un usuario.
- **Admin.html:** página solo accesible a administradores autenticados, se acceden a las diferentes acciones que puede realizar un administrador. Es decir, las páginas: list-users-admin.html y view-backups-admin.html.
- **list-users-admin.html:** página que permite ver una tabla con todos usuarios en el sistema y sus permisos. Así como dar permisos de administrador o quitárselos a cualquier usuario de la tabla mostrada. Es solo accesible para administradores autenticados.
- **view-backups-admin.html:** página que permite ver una tabla con información de los *backups* de todos los usuarios del sistema. Es solo accesible para administradores autenticados.

5.4.2 Aplicación cliente Android

En este apartado se describirán las clases contenidas en los diferentes paquetes creados para la aplicación Android. El fichero **google-services.json** se encuentra situado dentro de la carpeta androidransomware/app. Es un fichero necesario para el correcto funcionamiento de la recepción de notificaciones a través de Firebase.

- **com.albert.androidransomware.activities**: este paquete contiene todas las clases que son actividades y fragmentos. Es decir, las pantallas que el usuario ve e interacciona. Están formado por las siguientes clases:
    - **BackupAcitivity.java:** actividad que contiene los métodos relacionados con la creación del fragmento Backup.
    - **BackupFragment.java:** fragmento del Backup, añade la información a la interfaz de la vista detallada de un *backup.* También se encarga de gestionar las pulsaciones de los botones para modificarlo, eliminarlo o subirlo al servidor.
    - **BackupListActivity.java:** actividad que contiene los métodos relacionados con la creación del fragmento BackupList, es decir, la lista de *backups* que aparecen en la pantalla principal de la aplicación. Además de gestionar que sucede cuando seleccionas un *backup* para ver su información más detallada y poder operar con él.
    - **BackupListFragment.java:** fragmento de la lista de *backups*, contiene los métodos necesarios para actualizar la información gráfica de la lista de *backups*. También contiene la lógica del menú, con las opciones de añadir nuevo *backup*, ayuda, ajustes y cerrar sesión. En el caso de añadir nuevo *backup* se presenta un diálogo para que el usuario pueda introducir el nombre, descripción y frecuencia de subida. Luego puede seleccionar los archivos o ficheros que quiera. También se encarga de la lógica del RecyclerView para gestionar la carga de la lista de partidas en la pantalla móvil, solo cargando las que serán visibles y así optimizar el rendimiento y respuesta de la aplicación.
    - **ERPreferenceActivity.java:** actividad que se encarga de las preferencias locales del usuario. Guardando información como el número de móvil, el nombre de usuario, su ID único y el ID de Firebase.
    - **ERPreferenceFragment.java:** fragmento que carga las preferencias guardas en el dispositivo móvil en el fichero de ajustes.
    - **LoginActivity.java:** actividad que contiene los métodos necesarios para que el usuario pueda autenticarse e ingresar en la aplicación. El usuario introducirá su nombre de usuario, contraseña, *token* y número de teléfono. Y además se enviará el identificador único de Firebase para que el móvil pueda recibir notificaciones del servidor. Todo ello se enviará al servidor, el cual responderá y se mostrará un mensaje dependiendo de la respuesta. Si el *login* es incorrecto porque el nombre de usuario, contraseña o *token*, se mostrará el mismo mensaje que la aplicación web: "Nombre de usuario, contraseña o *token* incorrectos" (es una traducción ya que la interfaz de las herramientas diseñadas son en inglés al igual que los mensajes). Si el número de teléfono es incorrecto, se le comunicará al usuario.
    - **RecyclerItemClickListener.java:** actividad que se encarga de gestionar la lógica y funcionamiento del RecyclerView, particularmente al tocar los elementos de la lista en la pantalla.

    - **SplashActivity.java:** actividad que crea una animación con el logo y nombre del proyecto mientras se carga la aplicación.
- **com.albert.androidransomware.firebase**: este paquete contiene las clases necesarias para recibir y mostrar las notificaciones de Firebase.
    - **FirebaseIDService.java:** servicio que se encarga de capturar y guardar el ID de Firebase en las preferencias locales del usuario.
    - **MyFirebaseMessagingService.java:** clase que contiene la lógica para recibir las notificaciones de Firebase y procesarlas para que salga una notificación al usuario con la información pertinente.
- **com.albert.androidransomware.model:** este paquete contiene las clases del modelo de datos de la aplicación, así como la factoría del repositorio.
    - **Backup.java:** clase que modela un *backup*, con los campos necesarios como el título, el id, la ruta de archivos a copiar, la descripción, fecha de creación, frecuencia, fecha de última subida y si es la primera vez que se realiza la subida del *backup* al servidor.
    - **BackupRepository.java:** interfaz que contiene las funciones que describen las posibles acciones a realizar sobre la aplicación. Desde cargar la lista de *backups*, autenticarse, subir un archivo, modificar un *backup* o eliminarlo así como añadir un nuevo *backup*.
    - **BackupRepositoryFactory.java:** factoría que permite crear un repositorio del servidor (**ServerRepository**).
- **com.albert.androidransomware.server:** este paquete contiene las clases relacionadas con la conexión con el servidor. Incluyendo la librería VolleyMultipartRequest.
    - **ServerInterface.java:** clase que contiene los métodos de conexión con la API Rest del servidor. Utiliza la librería de Volley [37] para realizar las comunicaciones con el servidor y en el caso el envío de ficheros, utiliza la librería **VolleyMultipartRequest.** También utiliza un TrustManager para poder cifrar la comunicación y que sea a través de SSL
    - **ServerRepository.java:** implementa la interfaz de **BackupRepository**, contiene las llamadas a los métodos de **ServerInterface** y se encarga de gestionar los callbacks y los errores. Así como de recibir el archivo de respuesta en formato json y *parsearlo* para obtener la información acerca del *backup.* Por ejemplo al recibir la lista de *backups* de un usuario, debe recorrer la lista e ir creando un objeto Backup rellenándolo con la información que obtiene del archivo de respuesta.
    - **VolleyMultipartRequest.java:** librería que extiende las funcionalidades de Volley permitiendo el envío de ficheros. [38]

6 Integración, pruebas y resultados

Después de implementar el proyecto en base a los requisitos mencionados y atendiendo al diseño propuesto se han integrado los diversos módulos para comprobar su funcionamiento. La integración ha sido realizada por estadios, primero el servidor con la aplicación web. Posteriormente se ha unido con la aplicación cliente para Android.
A continuación se mostrarán las capturas de pantalla de la aplicación de las maquetas presentadas en la sección 4, tanto para la aplicación web como móvil. Para que se pueda apreciar la integración final del proyecto y el resultado del mismo.

Primero mostraremos las capturas de pantalla de la aplicación web. El navegador Google Chrome muestra un mensaje en la URL diciendo que la página no es segura. Esto es debido a que el certificado es autofirmado, y Chrome no lo reconoce como certificado de confianza. En la Figura 16 se puede observar la página de usuario con las diferentes acciones posibles. En la Figura 17 se puede ver la tabla con información de todos los *backups* del usuario.

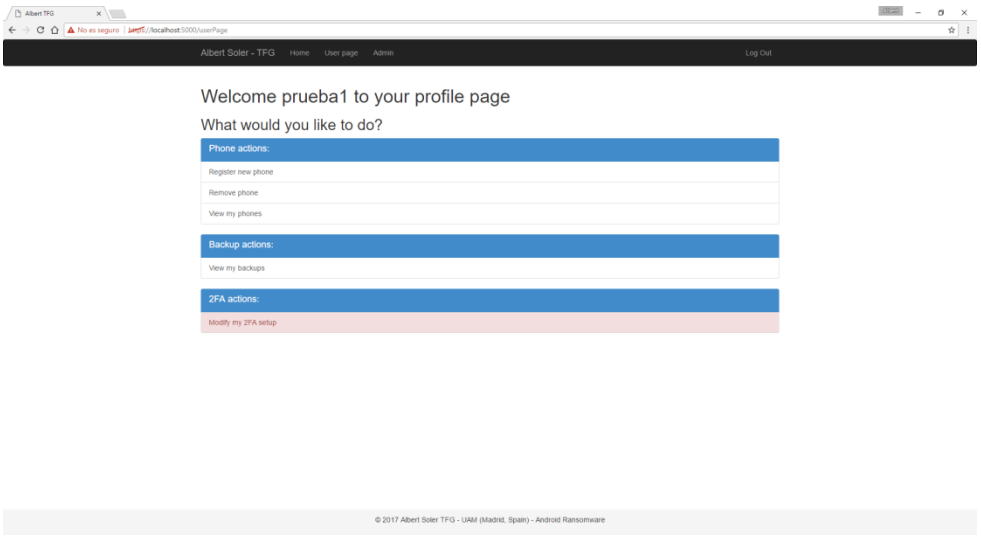

Figura 16: Captura de pantalla – Aplicación web – Página de usuario

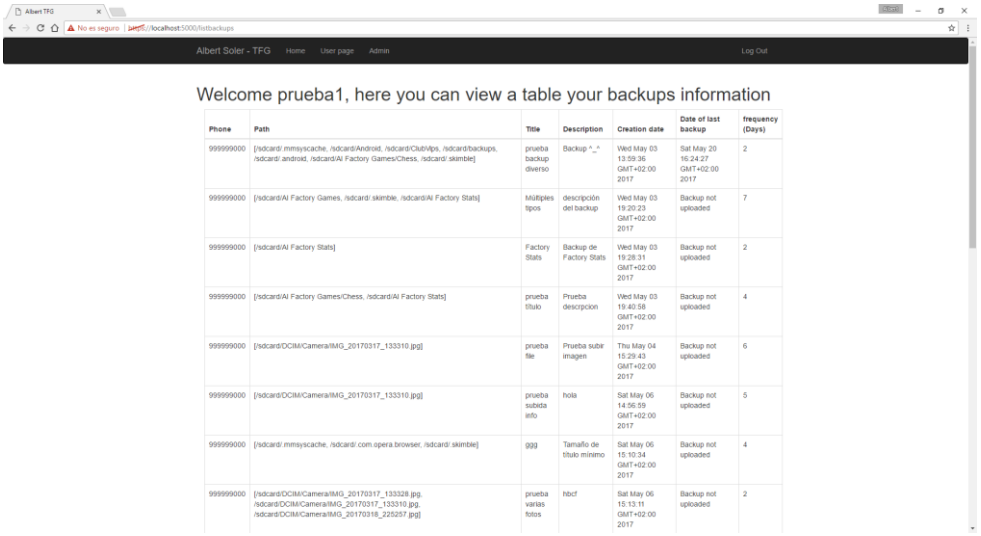

Figura 17: Captura de pantalla – Aplicación web – Ver todos los backups de un usuario

Ahora mostraremos las capturas de pantalla de la Aplicación cliente Android. En la Figura 18 se pueden ver la lista de *backups* con información del mismo como la ruta, fecha de creación y la descripción. Además en la parte superior se encuentra el botón para añadir un nuevo *backup* (la cruz verde), el menú de ayuda (la interrogación gris), y los tres puntos verticales que despliegan el resto del menú (ajustes y cerrar sesión).
En la Figura 19 se puede ver la pantalla cuando se ha seleccionado un *backup* que muestra información adicional como la frecuencia y la fecha de subida del *backup* al servidor. En este caso se ha pulsado el botón para subirlo al servidor y obtenido el mensaje confirmando que la subida del fichero ha sido satisfactoria. Se obtiene un mensaje por fichero, ya que el *backup* se sube fichero a fichero.
Por el contrario, en la Figura 20 podemos observar el mensaje de error al intentar subir un fichero con una extensión no aceptada. En ese caso, ese fichero no se subiría pero el resto sí.

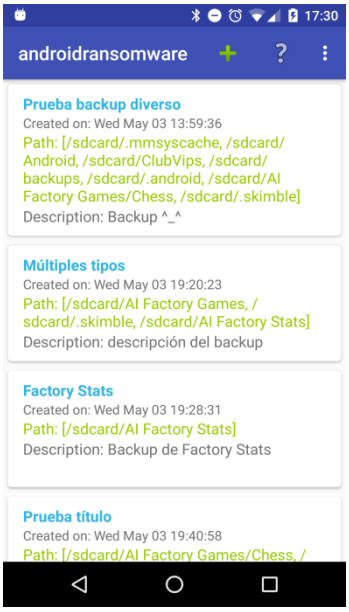

Figura 18: Captura Android – Lista de *backups*

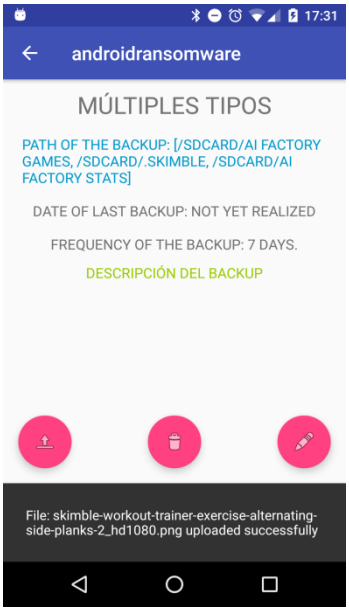

Figura 19: Captura Android – Subir fichero (satisfactorio)

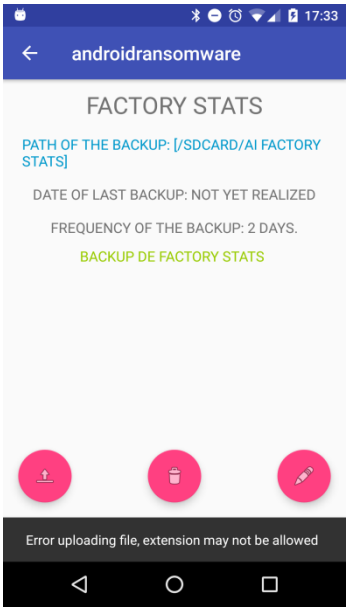

Figura 20: Captura Android - Subir fichero (error extensión)

Para verificar la corrección del sistema desarrollado se ha diseñado un plan de pruebas que se fueron realizando en paralelo con a la implementación del mismo. Las pruebas han sido adaptadas al tipo de problema, teniendo siempre presentes las peculiaridades del servidor, la aplicación web y la aplicación cliente para Android.

Se han realizado dos tipos de pruebas principalmente, las pruebas unitarias, cuya finalidad es comprobar que una función o módulo determinado cumplen su propósito sin tener en cuenta el resto. Para ello se ha probado los casos extremos y que nos devuelve error al introducir parámetros incorrectos en las funciones o API. Además se ha evaluado que el error devuelto y mostrado al usuario contenga toda la información necesaria para identificar el fallo.
El otro tipo de pruebas realizadas, son las pruebas de integración, dónde se prueba el funcionamiento del sistema en conjunto. Verificando la integración entre los diferentes módulos y componentes del mismo. Estás se han realizado probando las aplicaciones en su conjunto, conectado con el servidor las dos aplicaciones. Un ejemplo de ello son las figuras anteriormente mostradas (Figura 19 y Figura 20).

Para realizar las pruebas de la API REST se ha utilizado la herramienta Postman [45].

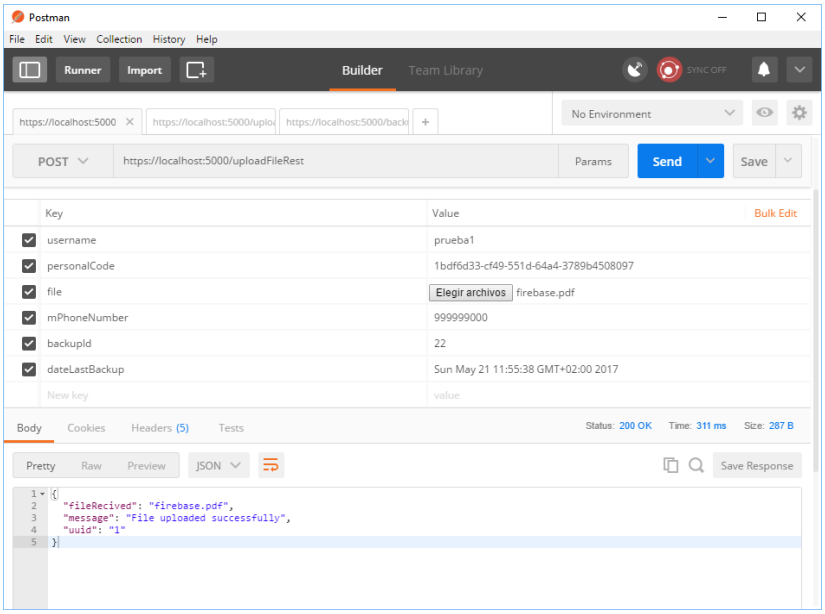

Figura 21: Captura de pantalla de la herramienta Postman – Subir archivo satisfactorio.

Como podemos ver en la Figura 21 se trata de una prueba de subir un archivo (un .pdf) al servidor simulando ser el usuario: prueba1. Como la subida ha sido satisfactoria el archivo tipo json que recibimos nos indica el fichero recibido, y el mensaje. Si la subida no fuera satisfactoria el campo "message" del json nos indicaría el motivo. Mostrando: "File not found" si no ha encontrado el fichero (ver Figura 22), "User not found" Si el usuario o ID único de éste es incorrecto (ver Figura 23) y "Error uploading file, extension may not be allowed" Si la extensión no está permitida o ha habido un error inesperado (ver Figura 24). El campo "uuid" también indica la existencia del error (número negativo) y el tipo de este.

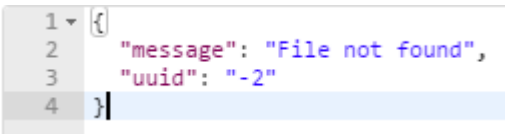

Figura 22: Postman – Error "File not found".

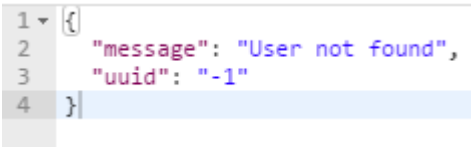

Figura 23: Postman – Error "User not found".

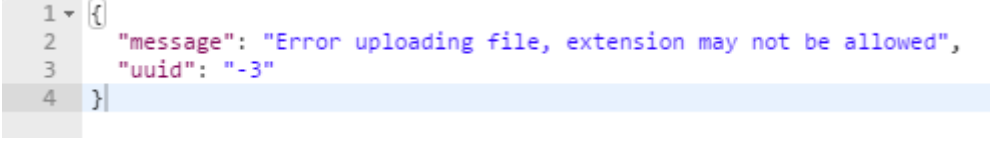

Figura 24: Postman – Error "Error uploading file, extension may not be allowed".

7 Conclusiones y trabajo futuro

Se ha podido desarrollar un sistema que mitigue el daño infligido por el *ransomware* destinado a dispositivos móviles Android. En concreto, una solución de copias de seguridad enmarcada en el seno de una empresa, que pueda ser integrada en un MDM. Es decir, fácilmente ampliable y escalable si fuera necesario.
Para la realización de este proyecto se han empleado los conocimientos adquiridos a lo largo de la carrera. Además, este trabajo fue seleccionado debido a un interés personal en el campo de la seguridad informática. Un campo de gran interés en la actualidad ya que conforme crece el número de amenazas y de dispositivos vulnerables, crece la demanda de profesionales de seguridad informática.
También se han perfeccionado los conocimientos en lenguajes de programación poco explorados durante el grado como Python, y Java orientado a dispositivos móviles. Además de aprender herramientas y *frameworks* nuevos como Flask, Mendeley e yED.
Destacar que el proyecto desarrollado ha sido un claro ejemplo del principio *security by design*. Es decir, teniendo en cuenta la seguridad desde el análisis y diseño del sistema. Y desarrollándolo con ello en mente, no como otros proyectos donde la seguridad es un añadido a posteriori.
Las competencias adquiridas tras la conclusión del trabajo son muy notables. Python es un lenguaje muy extendido y de amplia utilidad al ser multiplataforma. Android es el sistema operativo más utilizado del mundo [4] por lo que conocer cómo desarrollar aplicaciones para él es primordial. Además se están centrando muchos recursos en crear aplicaciones móviles dadas su alta disponibilidad y portabilidad.

7.1 Trabajo futuro

Después de evaluar las características y limitaciones del proyecto, se han pensado varias mejoras para aumentar la funcionalidad de la herramienta así como facilitar su uso.

Análisis de aplicaciones
La herramienta podría incorporar un botón para mandar el listado de aplicaciones instaladas en el dispositivo así como su hash al servidor. El servidor al recibir el listado podría consultar que los hashes correspondan a la aplicación y comprobar si es segura.

FileObserver
En vez de comprobar la fecha de modificación del fichero para evitar subir duplicados, se podría utilizar FileObserver [47]. Que permite obtener más información acerca de los ficheros como por ejemplo si han sido accedidos, si se ha cambiado su localización o si ha sido eliminado. Es una clase que permite hacer uso de un recurso del núcleo de Linux notificando cuando se producen cambios en los ficheros.
En relación con el RNF 10-eficiencia y consumo de batería de la aplicación Android-, FileObserver no consume batería ya que simplemente es notificada cuando ocurre un evento de tipo iNotify.

Porcentaje de subida al servidor de un *backup*
Para aumentar la claridad de la aplicación se podría implementar una notificación cuando se esté subiendo el *backup* que muestre su progreso. La Figura 25 es una maqueta de este concepto.

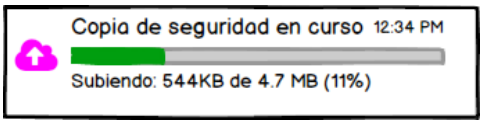

Figura 25: Copia de seguridad en curso

Recuperación automatizada de archivos desde un *backup*
Actualmente si un usuario quiere recuperar sus archivos de un *backup* debe ponerse en contacto con un administrador del sistema. Y el propio administrador le copiará los archivos desde el servidor al móvil. El problema que esto presenta es la necesidad de tener el móvil físicamente para recuperar los archivos y que es un proceso manual. Además, en el caso de necesitar recuperar diversos terminales puede convertirse en una tarea larga y tediosa.
Por lo tanto, una posible mejora sería que el usuario tuviera una opción desde su dispositivo móvil que le permita descargar los archivos de alguno de sus *backups*. La propia aplicación cliente Android se comunicaría con el servidor.

Política de *backup*
Una mejora sería dar la posibilidad al administrador o usuario de cambiar el tipo de copia de seguridad. En general, existen cuatro tipos diferentes de copias de seguridad [48]:

- Copia de seguridad completa: Se realiza un *backup* completo de todos los archivos del sistema.
- Copia de seguridad diferencial: Solo contiene los archivos que han cambiado desde la última vez que se hizo un *backup* de tipo completo [49].
- Copia de seguridad incremental: Se realiza un *backup* de todos los archivos modificados desde el último *backup* completo, diferencial o incremental.
- Copia de seguridad espejo: Es similar al *backup* de tipo completo, pero los datos no se pueden encriptar ni comprimir.

En este proyecto la primera vez que sea sube un *backup* al servidor es de tipo completo, y las siguientes subidas al servidor son de tipo incremental.

Es importante destacar que para protegerse adecuadamente frente al *ransomware*, en el servidor debe existir al menos una copia de cada fichero previo al ataque. Es lo que logramos combinando el *backup* completo y diferencial.

Detección de *ransomware*
Los requisitos de este TFG han venido dictados para desarrollar una herramienta de backup para posibilitar la recuperación del sistema frente a un ataque de *ransomware*. No obstante, y tal como se mostró en el capítulo 2.3 también existen técnicas orientadas a la detección del *ransomware* de modo previo a su despliegue. Éstas técnicas preventivas no han sido integradas en el actual prototipo, si bien se ha efectuado un primer análisis con Heldroid. Por tanto, la mejora futura de la herramienta desarrollada en este TFG involucra la inclusión de un módulo funcional para le detección de *ransomware* en el dispositivo móvil Android.

Referencias

[1] S. S. Response, "WannaCry ransomware," *Symantec Official Blog*, 2017. [Online]. Available: https://www.symantec.com/connect/blogs/what-you-need-know-about-wannacry-ransomware. [Accessed: 23-May-2017].

[2] bbc, "Cyber-attack: Europol says it was unprecedented in scale." [Online]. Available: http://www.bbc.com/news/world-europe-39907965. [Accessed: 23-May-2017].

[3] W. D. PythonResponder, "Microsoft Windows SMB Tree Connect Response denial of service vulnerability," *CERT Vulnerability Notes Database*, 2017. [Online]. Available: https://www.kb.cert.org/vuls/id/867968. [Accessed: 28-May-2017].

[4] J. M. Zuriarrain, "Android ya es el sistema operativo más usado del mundo," *El País - Tecnología*, 2017. [Online]. Available: http://tecnologia.elpais.com/tecnologia/2017/04/04/actualidad/1491296467_396232.html. [Accessed: 28-May-2017].

[5] A. Cavoukian, D. Ph, and M. Dixon, "Privacy and Security by Design : An Enterprise Architecture Approach," no. September, 2013.

[6] N. Andronio, S. Zanero, and F. Maggi, "HELDROID: Dissecting and detecting mobile ransomware," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015.

[7] F. Andronio, Niccolò and Zanero, Stefano and Maggi, "HelDroid: Dissecting and Detecting Mobile Ransomware," *Blackhat EU 2016, London*, 2016. [Online]. Available: https://github.com/necst/heldroid.

[8] K. Perkins, "Chapter 67 - Data Loss Protection A2 - Vacca, John R. BT - Computer and Information Security Handbook (Second Edition)," Boston: Morgan Kaufmann, 2013, pp. 1075–1092.

[9] A. Kharraz, S. Arshad, C. Mulliner, and W. Robertson, "UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware."

[10] R. Lipovský, L. Štefanko, and G. Braniša, "The Rise of Android Ransomware," pp. 1–19, 2015.

[11] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and Privacy in Social Networks*, 2013, pp. 197–223.

[12] HERB WEISBAUM, "Ransomware: Now a Billion Dollar a Year Crime and Growing," 2017. [Online]. Available: http://www.nbcnews.com/tech/security/ransomware-now-billion-dollar-year-crime-growing-n704646.

[13] R. Adnyana, "Ransomware Attacks The Cloud!," *Spin Backup*, 2016. [Online]. Available: https://spinbackup.com/blog/ransomware-attacks-the-cloud/. [Accessed: 28-May-2017].

[14] C. Everett, "Ransomware: To pay or not to pay?," *Comput. Fraud Secur.*, 2016.

[15] V. S. Net, "Los 'secuestros' de móvil atacan ya a uno de cada 100 teléfonos españoles," pp. 28–31, 2016.

[16] J. ALBORS, "Nuevos ransomware: el Virus de la Policía, ahora para Android," *We live security (ESET)*, 2014. [Online]. Available: https://www.welivesecurity.com/la-es/2014/05/06/nuevos-ransomware-virus-policia-ahora-para-android/. [Accessed: 28-May-2017].

[17] A. Dehghantanha, "Trends in Android Malware," vol. 8, no. 3, pp. 21–40.

[18] Sundar R, "Android.Lockdroid.E," *Symantec*, 2015. [Online]. Available: https://www.symantec.com/security_response/writeup.jsp?docid=2014-103005-2209-99.

[19] Marting zhang, "Android ransomware variant uses clickjacking to become device administrator," *Symantec Official Blog*, 2016. [Online]. Available: https://www.symantec.com/connect/blogs/android-ransomware-variant-uses-clickjacking-become-device-administrator.

[20] "TYPE_SYSTEM_OVERLAY," *Android Developers References*. [Online]. Available: https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#TYPE_SYSTEM_OVERLAY.

[21] Kellex, "Android Distribution Updated for April 2017," *droid-life*, 2017. [Online]. Available: http://www.droid-life.com/2017/04/06/android-distribution-updated-april-2017-nougat-closes-5/.

[22] N. Andronio, S. Zanero, and F. Maggi, "HELDROID: Dissecting and detecting mobile ransomware," in *Lecture Notes in Computer Science (including subseries Lecture Notes in*

Artificial Intelligence and Lecture Notes in Bioinformatics), 2015, vol. 9404.
[23] A. Ronacher, "Flask." [Online]. Available: http://flask.pocoo.org/. [Accessed: 11-May-2017].
[24] Python Software Foundation, "UUID objects according to RFC 4122." [Online]. Available: https://docs.python.org/2/library/uuid.html. [Accessed: 15-May-2017].
[25] et al Leach, "RFC 4122." [Online]. Available: https://tools.ietf.org/html/rfc4122.html. [Accessed: 15-May-2017].
[26] Google, "Material Design Guidelines." [Online]. Available: https://material.io/guidelines/material-design/introduction.html. [Accessed: 01-Jan-2017].
[27] Google, "Google Authenticator." [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl=en. [Accessed: 18-May-2017].
[28] Red Hat, "FreeOTP Authenticator." [Online]. Available: https://play.google.com/store/apps/details?id=org.fedorahosted.freeotp&hl=en. [Accessed: 18-May-2017].
[29] Lawrence Abrams, "TeslaCrypt and Alpha Crypt Ransomware Information," 2015. [Online]. Available: https://www.bleepingcomputer.com/virus-removal/teslacrypt-alphacrypt-ransomware-information. [Accessed: 18-May-2017].
[30] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proceedings - IEEE Symposium on Security and Privacy*, 2012, pp. 553–567.
[31] M. H. Eldefrawy, M. K. Khan, K. Alghathbar, T. H. Kim, and H. Elkamchouchi, "Mobile one-time passwords: Two-factor authentication using mobile phones," *Secur. Commun. Networks*, vol. 5, no. 5, pp. 508–516, 2012.
[32] M. Otto, "Bootstrap." [Online]. Available: https://getbootstrap.com/. [Accessed: 28-May-2017].
[33] Python Software Foundation, "Python 2.X vs 3.X," 2017. [Online]. Available: https://wiki.python.org/moin/Python2orPython3. [Accessed: 20-May-2017].
[34] P. Developers, "PiP," 2016. [Online]. Available: https://pypi.python.org/pypi/pip. [Accessed: 20-May-2017].
[35] SQLite Consortium, "SQLite." [Online]. Available: https://www.sqlite.org/. [Accessed: 20-May-2017].
[36] S. Cohen, "What challenges has Pinterest encountered with Flask?," 2016. [Online]. Available: https://www.quora.com/What-challenges-has-Pinterest-encountered-with-Flask/answer/Steve-Cohen?share=1&srid=hXZd. [Accessed: 21-May-2017].
[37] A. Developers, "Volley." [Online]. Available: https://developer.android.com/training/volley/index.html. [Accessed: 21-May-2017].
[38] Angga, "VolleyMultipartRequest," 2016. [Online]. Available: https://gist.github.com/anggadarkprince/a7c536da091f4b26bb4abf2f92926594. [Accessed: 21-May-2017].
[39] G. Developers, "Firebase Cloud Messaging (FCM)." [Online]. Available: https://firebase.google.com/docs/cloud-messaging/. [Accessed: 21-May-2017].
[40] GenyMobile, "GenyMotion," *2014*. [Online]. Available: https://www.genymotion.com/. [Accessed: 20-May-2017].
[41] Google, "Android Studio." [Online]. Available: https://developer.android.com/studio/index.html. [Accessed: 20-May-2017].
[42] R. Hipp, "SQLite Browser." [Online]. Available: http://sqlitebrowser.org/. [Accessed: 20-May-2017].
[43] Balsamiq, "Balsamiq Mockups." [Online]. Available: https://balsamiq.com/products/mockups/. [Accessed: 20-May-2017].
[44] YWorks, "yEd." [Online]. Available: https://www.yworks.com/products/yed. [Accessed: 20-May-2017].
[45] P. Technologies, "Postman," 2017. [Online]. Available: https://www.getpostman.com/. [Accessed: 21-May-2017].
[46] Mendeley Ltd, "Mendeley," 2017. [Online]. Available: https://www.mendeley.com/. [Accessed: 21-May-2017].
[47] Google, "FileObserver." [Online]. Available: https://developer.android.com/reference/android/os/FileObserver.html. [Accessed: 18-May-2017].
[48] S. Filippi, "Tipos de copias de seguridad," *internetlab*, 2013. [Online]. Available: http://www.internetlab.es/post/2104/tipos-de-copias-de-seguridad/. [Accessed: 28-May-

2017].
[49] W. Should, “C H A P T E R 1 Introduction to Backup and Recovery,” pp. 1–16, 2011.
[50] K. REITZ, “Virtual environment,” *The Hitchhiker’s Guide to Python.*, 2017. [Online]. Available: https://python-guide-pt-br.readthedocs.io/en/latest/dev/virtualenvs/. [Accessed: 21-May-2017].

Glosario

2FA Two Factor Authentication, autenticación en dos pasos. Aparte de la contraseña requiere otra información para poder iniciar sesión, como por ejemplo un Token.

2SV Two-step verification, verificación en dos pasos. Requiere que el usuario introduzca dos verificaciones diferentes para iniciar sesión. Ambos deberán ser de la misma categoría de lo contrario sería 2FA.

Android Sistema operativo enfocado principalmente a los dispositivos móviles con pantallas táctiles diseñado por Google y basado en el núcleo Linux.

API Application Programming Interface. Conjunto de procedimientos que ofrece una biblioteca para poder acceder a los recursos de un sistema.

APK Android Package Kit, es el formato del paquete de instalación y distribución de aplicaciones Android.

Bitcoin Criptodivisa y método de pago digital introducida a finales del 2008. Donde los pagos transcurren sin intermediarios y son almacenados públicamente, es pseudo-anónima.

Bootstrap Framework para HTML, CSS y JS para desarrollar proyectos *responsive* en la web.

Botnet Conjunto de ordenadores infectados que se ejecutan de manera autónoma y automática. Controlados por su artífice.

BSD (Licencia) Licencia de software libre permisiva otorgada principalmente para los sistemas Berkeley Software Distribution (un tipo de sistemas operativos Unix-like).

Cryptolocker Ransomware tipo troyano dirigido a ordenadores con el sistema Windows, que se popularizó a finales de 2013.

FCM Firecloud Cloud Messaging, Sistema de notificaciones de Google. Remplaza a GCM que era la versión anterior.

Flask Es un framework minimalista escrito en Python que permite crear aplicaciones web. Está basado en la especificación WSGI de Werkzeug y el motor de plantillas Jinja2 y tiene una licencia BSD.

Framework Conjunto de estructuras, herramientas y módulos que sirven como base para el desarrollo de software.

GPS	Global Positioning System, en castellano, Sistema de Posicionamiento Global es un sistema para determinar la posición de un objeto en toda la tierra.
HTTP	Hypertext Transfer Protocol, es un protocolo de comunicación que permite las transferencias de información en la www.
IMEI	International Mobile Station Equipment Identity, es un código que identifica unívocamente a un teléfono móvil.
Malware	Software maligno, tiene como objetivo dañar o infiltrarse en un sistema informático.
MDM	Mobile Device Management, tipo de software que permite administrar y monitorizar dispositivos móviles.
MoneXy	Solución de un monedero virtual basado en la nube.
MoneyPak	Tarjeta prepago.
OTP	One time password.
pbkdf2	Password-Based Key Derivation Function 2.
PIN	Personal Identification Number, es un número de identificación personal utilizado en ciertos sistemas como el teléfono móvil, para identificarse y obtener acceso al sistema.
QR Code	Es la evolución del código de barras, permite almacenar información en una matriz de puntos o en un código de barras bidimensional.
REST (API)	REpresentational State Transfer, es un tipo de arquitectura para el desarrollo web. Permite crear servicios y aplicaciones que puedan ser usadas por cualquier cliente o dispositivo que entienda HTTP.
Root (Android)	Permisos de superusuario.
SHA1	Secure Hash Algorithm 1, Hash criptográfico diseñado por la National Security Agency. Produce un mensaje de 20 bytes.
SMS	Short Message Service, Servicio de mensajería a través de texto utilizado en la mayoría de móviles.
SQLite	Sistema relacional de base de datos que encapsula todo el contenido de la base de datos en un único fichero a diferencia de los modelos cliente-servidor.
Token	Identificador único.
URL	Uniform Resource Locator, llamado de manera coloquial dirección web.

Anexos

A Planificación del proyecto

En este anexo mostraremos un cronograma con la planificación del proyecto.

1. Inicio del proyecto. Análisis del estado del arte, lectura de la documentación e investigación acerca del *ransomware* en Android. *10/10/2016.*

2. Análisis y diseño. Incluye la definición del proyecto, programación de hitos, análisis de requisitos y diseño. *22/11/16.*

3. Desarrollo del proyecto, codificación y pruebas.
   A Investigación sobre HelDroid. *30/11/2016.*
   B Investigación sobre 2FA en Flask. 10/12/2016.
   C Desarrollo del primer prototipo del servidor. 13/01/2017.
   D Añadir SSL al servidor. 15/01/2017.
   E Comienzo del desarrollo de la aplicación web. 20/01/2017.
   F Segundo prototipo del servidor y aplicación web. 20/02/2017.
   G Comienzo de la aplicación Android. 23/02/2017.
   H Pruebas unitarias del servidor. 03/03/2017.
   I Implementar Firebase. 10/03/2017.
   J Pruebas de integración de la web y el servidor. 14/03/2017.
   K Desarrollo de la API REST. 18/03/2017.
   L Desarrollo de la aplicación Android y primer prototipo de la misma. 02/04/2017.
   M Codificación completa de la aplicación Android. 09/04/2017.
   N Codificación completa de la aplicación web y servidor. 20/04/2017.
   O Pruebas de integración entre los tres módulos. 02/05/2017.

4. Elaboración de la memoria
   A Redacción de la misma. 13/05/2017
   B Revisiones y finalización de la memoria. 29/05/2017

5. Finalización del proyecto. 30/05/2017

B Manual de instalación

Servidor Flask:

Para instalar el servidor Flask solo es necesario seguir estos pasos:

Instalar (vía pip[34]) *virtual enviroment* [50]

```
$ pip install virtualenv
```

Crear un *virtual enviorment:*

```
$ cd my_project_folder

$ virtualenv my_project
```

Activarlo

```
$ source my_project/bin/activate
```

Importar todas las dependencias del proyecto

```
$ pip install -r requirements.txt
```

Ejecutar la aplicación

```
$ python app.py
```

Conectarse con la aplicación a través del navegador en la URL: `http://localhost:5000`

Certificado

Para crear el certificado simplemente deberemos ejecutar (doble click) el fichero: `scriptCert.sh` y rellenar la información necesaria.
Una vez creados moverlos a una carpeta llamada `resources`

Aplicación Android

Para instalar la aplicación Android simplemente abrir la aplicación en Android Studio y darle al play para instalar el APK en el dispositivo o emulador (el emulador debe tener los servicios de Google Play para que funcionen las notificaciones)